Genta

Redefining project management

Software Engineering Major Project

Yav Find for David Noé

Jump to Table of Contents

Table of contents

TABLE OF CONTENTS` 2

PROJECT LINKS 4

TIMELINE 5

October 23, 2024 5
October 30, 2024 5
November 3, 2024 7
November 27, 2024 7
Nov 30, 2024 9
Dec 19, 2024 9
Jan 18, 2025 9
Jan 20, 2025 9
Feb 11, 2025 10
Feb 18, 2025 10
Feb 22, 2025 10
Feb 25, 2025 10
Mar 7, 2025 11
Mar 8, 2025 11
Mar 10, 2025 12
April 18, 2025 12
April 23, 2025 12
April 24, 2025 13
April 25, 2025 – April 30, 2025 (untracked) 13
May 1, 2025 13
May 2, 2025 14
May 3, 2025 14
May 4, 2025, ⭐ 14
May 5, 2025 14
May 9 – May 12, 2025 (semi-tracked) 15
May 14, 2025 15
May 25, 2025 15
May 26, 2025 16
May 27, 2025 16
May 28, 2025 16
May 29, 2025 16
May 30, 2025 16
June 3, 2025 17
June 4, 2025 17
June 5, 2025 17
June 7, 2025 17
June 8, 2025 17

June 9 – June 10 [3:15am] *18*

SOFTWARE DEVELOPMENT APPROACH **6**

SCHEDULING AND TASK ALLOCATION **8**

SOCIAL, ETHICAL, AND COMMUNICATION ISSUES WITH COLLABORATIVE WORK **12**

DEFINE AND ANALYSE THE PROBLEM REQUIREMENTS **16**

NEEDS AND OPPORTUNITIES **16**

BOUNDARIES **16**

FEATURE LIST [BACKLOG] **17**

IDENTIFICATION OF APPROPRIATE SOFTWARE DEVELOPMENT TOOLS. **18**

SITUATION 18
TOOL 18
REASON 18

SOFTWARE APPLICATIONS **25**

IMPLEMENTATION METHOD **27**

PROBLEM CONTEXT **30**

QUALITY ASSURANCE, QUALITY CRITERIA, COMPLIANCE, AND LEGISLATIVE REQUIREMENTS **31**

FINALISED IMPLEMENTATION **33**

SYSTEMS MODELLING **35**

UI DESIGN + STORYBOARD **35**

JSON OUTLINE 41
Schema *45*
Database data dictionary *45*
API ROUTES AND GENERAL SERVER HANDLING. 47
PSEUDOCODE 49
https://genta-api.online> server pseudocode *49*
https://genta.live > Server pseudocode *57*
https://genta.live > served JS pseudocode *58*
DESK CHECKING 62
DATA FLOW DIAGRAM 64
DATA DICTIONARY 65
*genta-api.online / **getUserDataFromDb()*** *65*
*genta-api.online / **getData ()*** *66*
*genta-api.online / **editDatabase ()*** *66*
*genta-api.online / **updateData ()*** *67*
*genta.live / **all*** *68*
CLASS DIAGRAMS (NOT APPLICABLE TO MY PROJECT) 68
STRUCTURE CHARTS 69
DECISION TREES FOR **GENTA API** 70

DECISION TREES FOR **GENTA CLIENT** 71
SOLUTION TO SOFTWARE PROBLEM **72**
VERSION CONTROL **93**
CODE OPTIMISATION **93**
TESTING METHODS **95**
TEST DATA TABLES **97**
BOUNDARY TEST 97
PATH TEST 98
METHODOLOGY TO TEST AND EVALUATE CODE **99**
ANALYSIS OF SOLUTION AGAINST QUALITY SUCCESS CRITERIA **100**
ANALYSIS OF FEEDBACK **102**

Project links

Item	Description	Hyperlink
Figma	Genta's **UI design** was built on Figma. The Figma also contains aspects of the documentation and the video.	Open Figma ↗
Github Repo for ***Genta API***	Genta's **backend microservice code** is stored on Github. This is referred to as Genta API	Open Github Repo ↗
Github Repo for ***Genta General***	Genta's **frontend microservice code** is stored on Github. This is referred to as Genta General	Open Github Repo ↗
Deployed Backend	As per my client's requirements, Genta was deployed. **The frontend fetches its data from this address**; the backend microservice.	Open Genta API ↗
Deployed Frontend	As per my client's requirements, Genta was deployed. This link goes to the final, deployed frontend, **the link that end users go to in order to use Genta.**	Open Genta ↗

(No longer deployed as of 3/7/25)		

Timeline

Begins October 23, 2024, and ends July 12, 2024.

Timeline is redacted for privacy.

Research and planning

Software development approach

Research and use the Waterfall, Agile, and WAgile software development approaches.

Waterfall

According to Atlassian (n.d.)[1], waterfall is a project management methodology where tasks are completed in a strict cascading order. Once one task is finished, you have to move on to the next task and cannot go back, similar to a waterfall in which water never flows backwards.

Figure 1 An example of a Waterfall project. Atlassian (n.d.)

A typical waterfall methodology consists of 5 parts; (1) defining project requirements, (2) designing the project itself and the roadmap to develop it, (3) send the design to developers (4) verify if the product reaches its requirements and passes quality assurance checks, and (5) maintain the product post release to fix any bugs and add new features

The waterfall methodology is more attractive to project managers because its rigidity leaves little room for confusion. Also, due to Waterfall's strictly linear progression it is easier to track the progress of a project overtime through measuring it against predefined milestones and deliverables. However, the design of waterfall can be problematic, as the rigidity means that a team cannot move forward until all problems are fixed and can't move back to resolve issues with the previous release. In the real world, this can lead to delays and cost increases, especially if a project is more complex and dynamic in nature.

Agile

Laoyan (2024)[2] describes how Agile isn't a framework, instead it's a mindset, in which the goal is to move quickly and efficiently while upholding the four pillars of Agile;

1. Valuing teamwork and collaboration rather than trying to follow a set of rules

[1] Atlassian (n.d.) Waterfall Methodology, Accessed Jan 18 2025, https://www.atlassian.com/agile/project-management/waterfall-methodology

[2] Laoyan, S (2025) What is Agile methodology? (A beginner's guide) Accessed Jan 18 2025 https://asana.com/resources/agile-methodology

2. Value **a functional product over a well-documented one,** (*does not mean no documentation*)
3. Focus on collaborating with the customer instead of contract negotiation.
4. Respond to change instead of trying to follow a plan. **This is a key advantage compared to waterfall.**

There are 12 principles of Agile, however I've gone through and recognised how each one works with my project

- Agile shows that I should **continuously deliver** to my client, which means that during development I should strive to make one aspect functional so they're able to interact with it and provide their thoughts early. This also lets them be more engaged with the development of the project.
- **Embracing change** even if it is late in the project is key to Agile. The only metric that matters is a working end product.
- Agile highly recommends **face-to-face** communication over all else. This means regularly calling my client instead of going for transient DMs. This also means that I should set up a clear schedule.
- Working at a pace that **minimises burnout** is crucial, especially considering what happened during Fresher.
- **Prioritise simplicity** instead of overcomplicated solutions to problems.

Agile as a mindset has many different ways of implementation, like Kanban for visual management, Scrum for small teams, Extreme Project Management (XPM) for complex projects, and more.

WAgile

Jaworski (2024)[3] describes WAgile as a hybrid project management methodology that combines waterfall and agile, taking the best parts of both. Waterfall's rigid cascade of tasks makes it predictable, clear, and easy, but it also makes it inflexible, risky and slow. Agile solves these problems by being more flexible, customer focused, fast and efficient (*agile*). However Agile can result in scope creep and is unpredictable.

As explained by Cloud Coach (2019)[4], WAgile exists due to the unrealistic expectations of conforming to a single methodology, and teams should customise the project

[3] Jaworski, B (2024) A guide to WAgile project management, Accessed Jan 18 2025 https://blog.logrocket.com/product-management/wagile/

[4] "Cloud coach" (2019) What is WAgile Project Management, Accessed Jan 18 2025 https://cloudcoach.com/blog/what-is-wagile-project-management/

management solution based on their needs. An implementation of WAgile for a team that maintains existing software could be that the release roadmap is waterfall, but when a new problem is introduced, they can tackle it with Agile.

What I've gone with

While researching the differences between certain agile frameworks, I've seen a repeating pattern in which teams are not following a single framework but using hybrid models that work better for them. "*...your decision doesn't need to be so black and white. Hundreds of teams are using hybrid models influenced by both scrum and kanban.*" (Kehkopf, n.d.)[5].

For now, I'm going to use Waterfall to complete all planning parts of the project, like identification and idea generation, and then once it gets to UI design and development, I'll implement an Agile approach. This is expanded on further below.

Scheduling and task allocation

Apply project management to plan and conduct the development and implementation of a project and software engineering solution.

The project can be split into different aspects that can be managed via Waterfall and/or Agile. In terms of collaborative tools used, I will be using Github to handle my projects.

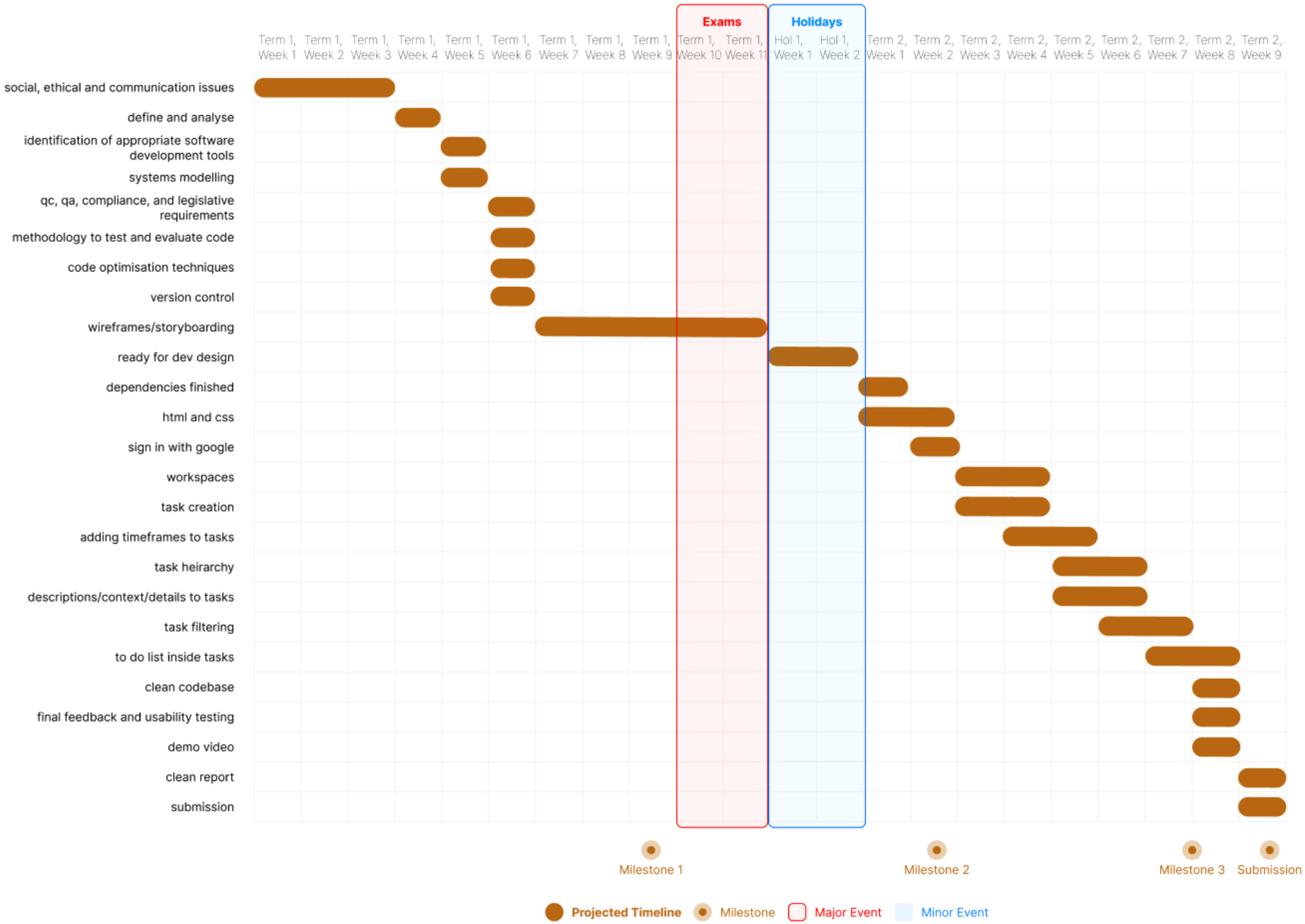
Waterfall	Agile
Social, ethical, and communication issues with collaborative work. *Completed Feb 18, 2025.* *This should be completed as soon as possible to ensure that communication between me and my client is as effective as possible.* Define and analyse problem requirements Define problem context Define needs and opportunities Define boundaries *Completed Feb 25, 2025.* *These should be complete so that the design is focused and properly targets the client's requirements*	

[5] Rehkopf, M (n.d.) Kanban vs Scrum, which agile are you? Accessed Jan 18 2025 https://www.atlassian.com/agile/kanban/kanban-vs-scrum ΩΩz≈ttttt

Identification of appropriate software development tools, including selection of tools that are going to be used in the UI/UX design and development of the project. *Completed Mar 8, 2025.* *This should be complete so that I can start designing and developing with the appropriate tools*	Can begin UI/UX Design *Begun April 18, 2025 - Ended April 23, 2025* *Once a clear outline of the software solution is created, I'm able to begin developing the UI/UX design, however this can run in tandem with the agile tasks.*
Quality criteria, compliance, and legislative requirements *Completed Mar 10, 2025.* *This should be complete so that I have a good baseline to go off of.* Systems modelling *Completed May 14, 2025.* *Understanding and modelling a system is crucial to build an effective solution.* Version control *Completed May 14, 2025* *Create a system to manage versions of the system using Git.*	Can begin Software development with quality Assurance *Begun May 25, 2025 - Ended June 9, 2025* *Once I've created a way to properly track progress overtime, I'm able to start my software development. This may happen in a package in which each iteration has a complete package of tools*
Methodology to test and evaluate code *Completed June 8, 2025* *Developing a clear methodology to test and evaluate the code is necessary for seamless development* Code optimisation techniques *Completed June 9, 2025* *Code should be written with optimisation in mind, so understanding it is essential* Evaluation of solution	

Completed June 10, 2025 Analysis of solution against quality success criteria *A final evaluation of the software solution is required but can only happen at the end of development.*	

GENTA **MAJOR**


Exams
Holidays
Term 1, Week 1
Term 1, Week 2
Term 1, Week 3
Term 1, Week 4
Term 1, Week 5
Term 1, Week 6
Term 1, Week 7
Term 1, Week 8
Term 1, Week 9
Term 1, Week 10
Term 1, Week 11
Hol 1, Week 1
Hol 1, Week 2
Term 2, Week 1
Term 2, Week 2
Term 2, Week 3
Term 2, Week 4
Term 2, Week 5
Term 2, Week 6
Term 2, Week 7
Term 2, Week 8
Term 2, Week 9
social, ethical and communication issues
define and analyse
identification of appropriate software development tools
systems modelling
qc, qa, compliance, and legislative requirements
methodology to test and evaluate code
code optimisation techniques
version control
wireframes/storyboarding
ready for dev design
dependencies finished
html and css
sign in with google
workspaces
task creation
adding timeframes to tasks
task heirarchy
descriptions/context/details to tasks
task filtering
to do list inside tasks
clean codebase
final feedback and usability testing
demo video
clean report
submission
Milestone 1
Milestone 2
Milestone 3
Submission
Projected Timeline
Milestone
Major Event
Minor Event

Social, ethical, and communication issues with collaborative work

Explore social and ethical issues associated with project work, including working individually, collaboratively, and responding to stakeholders. || Explore communication issues associated with project work.

According to Brodowicz (2024)[6] social issues of project work are sourced from the people and the communication between them throughout the entirety of the project. Brodowicz states that, while such issues can cause immediate problems for productivity in the project, it can lead to much larger issues that can affect the entire project and even extend beyond it.

Individual

[Social] Project work on the individual level can cause problems, such as social isolation. Tulane University (2020)[7] explains that while individual work can be a good thing, individuals in too much solitude can cause social isolation. These people can experience depressed moods, low self-esteem, and are more prone to anxiety. Furthermore, especially in educational contexts, people working individually have a more difficult time accessing immediate and responsive feedback. This can cause a narrower mindset, hinder progress, and result in errors to be overlooked over long periods of time (Verma, 2022)[8].

2 "Eml" (2020) If working from home is here to stay, what are the risks? Accessed 11 Feb 2025, https://employeemanagement.co.uk/if-working-from-home-is-here-to-stay-what-are-the-risks/

As shown earlier, due to social isolation, there are missed opportunities for external feedback and validation of ideas, process, and output. By extension, this means that problem solving can be significantly less efficient as it's limited to an individual's perspective. To solve this, individuals need to develop clear communication skills and actively seek feedback from peers and supervisors. An organisation facilitating individual work also needs to provide a framework to ensure responsive and accurate feedback.

---

[6] Brodowicz, M (2024) Social and Organisational Issues in Project Management, Accessed Feb 11 2025, https://aithor.com/essay-examples/social-and-organisational-issues-in-project-management

[7] Tulane University (2020) Understanding the Effects of Social Isolation on Mental Health, Accessed Feb 11 2025, https://publichealth.tulane.edu/blog/effects-of-social-isolation-on-mental-health/

[8] Verma, S (2022) 5 Reasons Why Immediate Feedback is Important for Effective Learning, Accessed Feb 11 2025 https://www.blog.intedashboard.com/blogs/tbl-learning/immediate-feedback

There are also ethical issues to working individually. For example, working alone reduces the ability for an organisation to enforce transparency and ethical code of conduct. Also, due to not having any external feedback available to them, individuals could introduce bias into their work, manifesting overtime if no system is in place to handle them. With time, poor quality work can also manifest due to the lack of external influence, (Frohnhoefer, 2023)[9].

Collaborative

Collaborative work solves some issues of individual work, like having a support system around individuals and each individual being able to be better equipped to deal with roadblocks. However, the nature of collaborative work also can prove problematic at times. For example, Berwicks (2025)[10] shows that in collaborative settings, team members can be prone to supress dissenting opinions about topics in order to speed up the process of consensus. This means that teams can often leave out important details and alternate perspectives. Using strategies, like assigning a "Devil's advocate" to directly challenge assumptions and ideas, can help alleviate this problem.

Furthermore, within collaborative work settings, people of different ethnicity, gender, and background, may be present in a single team. When diversity issues, like microaggressions, assumptions, and prejudice are handled improperly, diverse teams can face imbalances in workload across team members, may be excluded, and may feel undervalued compared to other team members ("HackingHR", 2024)[11].

3 Hendricks, A (n.d.) How to manage stakeholder expectations. Accessed 11 Feb 2025 https://simplystakeholders.com/manage-stakeholder-expectations

In an ethics standpoint, collaborative work can be prone to a lack of proper acknowledgment to those who have done more or less than others. Furthermore, as shown earlier, collaborative dynamics can cause individuals in a group to avoid dissent and hence take actions against their morals in order align with consensus. Hence, this can cause result in

[9] Frohnhoefer, R (2023) Making Ethical Decisions in Challenging Project Situations, Accessed Feb 11 2025 https://blog.iil.com/making-ethical-decisions-in-challenging-project-situations

[10] "Berwicks" (2025) Breaking the consensus trap: How to avoid groupthink in the workplace. Accessed Feb 11 2025 https://www.bbcdigital.com.au/breaking-the-consensus-trap-how-to-avoid-groupthink-in-the-workplace/

[11] "HackingHR" (2024). Breaking Barriers: Inclusive Communication Across Cultures and Genders, Accessed 11 Feb 2025 https://hackinghrlab.io/blogs/inclusive-communication/

unethical practices, like cutting corners and overlooking ethical issues Berwicks (2025)[12].

Naybour (2022)[13] shows that due to the tight-knit nature of collaborative work, these projects often involve complex on ongoing exchanges, and minor cases of misinformation and misunderstandings can manifest into larger problems rapidly. This can be exacerbated when the roles and responsibilities of team members are not clearly defined, documentation is poorly detailed, and ineffective use of communication and collaboration tools.

Stakeholders

In terms of responding to stakeholders, individual stakeholders often have varying levels of power and influence, and it is common for their interests to not align. In a sense, stakeholders are subject to the same social fallacies in collaborative work that the base team has, in which stakeholders could avoid dissent to improve consensus and be subject to power imbalance due to diversity ("Community tool box" n.d.)[14]. However, due to the nature of stakeholders and their purpose, the needs of a stakeholder can be influenced by external or internal factors. These factors could result in stakeholders challenging a process based on its outcome, damaging relationships between the team, organisation, and stakeholder. To solve this, teams and organisations need to closely manage stakeholder expectations (Hendricks, n.d.)[15].

As stakeholders have power over a team, stakeholders arguably are the most prone to ethical issues. According to Machado (2024)[16], unethical communication, such as misrepresenting information between a stakeholder and a team and downplaying problems can erode trust and cause skewed expectations for a project. Stakeholders have the responsibility to collect data about individuals in a

4 Large Language Models (LLMs), packaged in consumer-friendly text generation tools like Google's Gemini and Anthropoic's Claude AI, have warnings that generated content may not always be accurate.

[12] Berwicks (2025) Breaking the Consensus Trap: How to Avoid Groupthink in the Workplace, Accessed 18 Feb 2025. https://www.bbcdigital.com.au/breaking-the-consensus-trap-how-to-avoid-groupthink-in-the-workplace

[13] Naybour, P (2022) Poor communication: cause or symptom in project failures? Accessed 18 Feb 2025. https://www.apm.org.uk/blog/poor-communication-cause-or-symptom-in-project-failures

[14] "Community tool-box" (n.d.) Identifying and analysing stakeholders and their interests. Accessed 11 Feb 2025. https://ctb.ku.edu/en/table-of-contents/participation/encouraging-involvement/identify-stakeholders/main

[15] Hendricks, A (n.d.) How to manage stakeholder expectations. Accessed 11 Feb 2025 https://simplystakeholders.com/manage-stakeholder-expectations

[16] Machado, M (2023) Project Management Ethics: Importance of Integrity and Accountability, Accessed 11 Feb 2025 https://instituteprojectmanagement.com/blog/project-management-ethics-importance-of-integrity-and-accountability/

team in a non-invasive way, like always seeking informed consent and ensuring that the data is confidential and anonymous. Furthermore, when a project involves innovation, stakeholders need to assess the social impacts the project may have and ensure that ethical values of the project are met. For example, Artificial Intelligence (AI) tools, which have had a remarkable societal impact, have warnings to notify users to not use generated content as factual evidence without proper scrutinisation.

Communication issues between stakeholders are also present, especially when misinterpretations occur between the stakeholder and the team, resulting in projects that fail to meet the stakeholder's expectations, needs, and requirements. For example, a team finds that stakeholder expectations are unrealistic, but cannot communicate that with the stakeholder, and hence the team fails to deliver a sufficient project (Hendricks, n.d.)[17]. Communication barriers can arise as a result of cultural issues from diverse team members/stakeholders, a mismatch in communication styles between stakeholders (i.e. contrasting formality in report writing vs informality in conversations), and even a general lack of empathy, respect, and patience (Gratis, 2024)[18].

[17] Hendricks, A (n.d.) Top 5 common mistakes in managing stakeholders and how to avoid them, Accessed 18 Feb 2025, https://simplystakeholders.com/top-5-common-mistakes-in-managing-stakeholders-and-how-to-avoid-them/

[18] Gratis, B (2024) Overcoming language barriers to communication, Accessed 18 Feb 2025 https://nulab.com/learn/collaboration/overcoming-language-barriers-communication/

Identifying and defining

Define and analyse the problem requirements

Project management solutions are often built to facilitate teams, enhancing existing team management methodologies, like Agile, Scrum, and Kanban. But what happens when you're just one person? With over 1.5 billion freelancers worldwide, completing projects individually is more common than you think. Yet there is no one solution to effectively plan, scope, and manage projects at an individual level.

Needs and opportunities

Individuals who work independently, like freelancers, solo entrepreneurs, and researchers, lack effective project management tools that are tailored to their needs. Furthermore, traditional project management methodologies are often too complex and/or too team oriented for individuals. People who work independently **need** an easy way to track tasks, manage their time, and stay organised without having to navigate complex team-based project management methodologies. Hence, there is an **opportunity** to create a tool that empowers independent workers to be more organised and efficient.

Genta **addresses the need** to create a more simplified, and more accessible project management approach for individuals. As such, Genta takes the **opportunity** to fill a gap in the market by offering a project management solution that is specifically tailored to individuals, removing the complexity of team-based project management.

Boundaries

In order of rigidity

- No AI features, maintaining simplicity and allows us to focus on core functionality.
- Because Genta is meant for individuals, there should be no collaboration features.
- Genta will avoid trying to rigidly adhere to a single project management framework and instead focus on a more flexible, individual-centric workflow.
- Genta will lack note taking features, avoiding turning into a tool like Notion.
    - Instead Genta might have small text fields to give more context to the task.
    - Users WILL be able to link resources to a specific task.
- To keep the user in control of their work: Genta to not turn into a micromanagement tool, Genta's workflow resemble more of a progress tracker and organiser tool. This also avoids individuals procrastinating their work by trying to plan.

Some other boundaries about performance include:

- Sign in with Google only. This is to ensure that there are no security issues, especially when storing passwords and usernames at rest.
- Desktop-priority design. Mobile should not limit desktop, and if accommodations for mobile are too large, it is not worth dealing with mobile.

Feature list [Backlog]

Genta is a tool to track goals and set milestones, allowing users to scope out their project tin a way that works for them

MVP [Minimum viable product]:

- Basic task functionality
    - Being able to distinguish between task, deadlines, and meetings
    - Basic level of filtering tasks
    - Well established task hierarchy, allowing subtasks.
    - Adding descriptions, context, and details to tasks
    - Adding timeframes for a set of tasks
    - To do list inside of tasks, allowing users to track granular tasks
    - Plaintext notes section
- Workspaces
    - Workspace setup system including deadlines and an overall description of the project/workspace.

Post-MVP: (in order of priority)

- Links in events
- To-do list on the sidebar
- Gannt chart creator
- Statistics, allowing users to see how they are progressing with their project
- Links embedded into tasks
- File uploading
- Mobile view

Tools to develop ideas and generate solutions

Identification of appropriate software development tools.

Explore tools used to develop ideas and generate solutions. Investigate types of software implementation methods.

Situation	Tool	Reason
Brainstorming, mind-mapping, and storyboards	Miro	**Pros** - One place for UI design research and team collaboration - Allows for project management **Cons** - Very team focused - Only 3 free boards - No education plan - It is marketed as an "everything" tool, so it might not be the best option for just brainstorming **Applicable?** - Some aspects of the tool *are* applicable, but a lot of it aren't.
	Figma	**Pros** - I'm used to this. - First choice by a lot of large businesses, like Google and Microsoft. - Has education options, allowing me to use it as an education system. **Cons** - Not inherently part of the school **Applicable?** - It is not a fully perfect solution, but it fits the requirements, and our needs.
	MS Whiteboard	**Pros** - Open space to draw anything - Amazing for brainstorming, mind-mapping, and BASIC storyboarding. **Cons** - Neither me or my client use it enough - While it is cross platform, there is minimal integration into other applications **Applicable?**

		- Yes.
	Paper	**Pros** - Very easy to use - Very natural **Cons** - No online sync. - Cannot export as PNG or PDF. - Can get lost. - Lots of paper can be hard to manage. **Applicable?** - Definitely.
Data dictionaries	Excel	**Pros** - Makes everything structured - Relatively easy to input information **Cons** - May not be the best option for a data dictionary - Not very visually friendly - Restricted amount of info that can be put into a cell until it looks like too much to read. **Applicable** - To a limited extent
	Word	**Pros** - Same tool I am using for my project - I have made data dictionaries for my past 3 software projects in word - Tables look elegant and neat **Cons** - Manual data entry **Applicable** - Yes
	Data documentation tool like dbdocs	**Pros** - Easy and quick to use - Script to take all variables out of program and generate a data dictionary **Cons** - Exceeds the scope of the project - Typically used for databases **Applicable** - Yes in the real world, no for the HSC.
Algorithm design	Draw.io (and other flowchart makers)	**Pros** - Great for flowcharts - Easy to use and export - Not many restrictions on what you can/can't do

		Cons - I will be using pseudocode more often compared to flowcharts **Applicable** - Yes
	Figma	**Pros** - A lot of flexibility - I have used it a lot - Version control and very easy sync **Cons** - Can be a bit too flexible - Not easy to use for flowchart creation **Applicable** - Yes, but not the best option
	Word	**Pros** - Great for word processing, hence it is a great for pseudocode **Cons** - Very difficult to create flowcharts on word. It is possible but too tedious and not worth the time. **Applicable** - Yes, for pseudocode. No for flowcharts.
Code generation	VS code	**Pros** - I am used to it - It has worked for all my projects - Community standard - Better onboarding experience **Cons** - Automatic venv startup for Flask is unreliable. **Applicable** - Very
	Visual studio	**Pros** - More robust, and is a proper IDE - Does not require the installation of extra things **Cons** - More heavy - I do not think I need the extra features **Applicable** - Yes, but overkill
	Sublime Text	**Pros** - Nicer to use when workflow is set up correctly - Web development is possible **Cons**

		- No onboarding - Does not help the user with extensions (you need to look for them yourself) - Not a community standard **Applicable** - Yes, not ideal
	Notepad++	**Pros** - Light weight - Simple to use **Cons** - Not feature rich enough - Windows only **Applicable** - Possible, but not very realistic
Testing and debugging	Unittest *pkg* (Python, 2025)[19]	Pythons built in framework for unit testing **Pros** - No extra installation - Good for basic unit testing **Cons** - Boilerplate heavy - Not feature rich **Applicable** - Yes, but not to an unlimited extent
	Pytest *pkg* (Pytest, n.d.)[20]	Testing framework for python **Pros** - Concise and readable syntax - Good for unit and integration testing - Easy Flask integration **Cons** - Requires installation - Somewhat steep learning curve **Applicable** - Yes
	Flask-testing *pkg* (Flask, n.d.)[21]	Testing extension specifically for Flask applications **Pros** - Easy to send requests to Flask routes - Simplifies setup

[19] Python [Documentation] (2025) unittest – Unit testing framework, Accessed March 10, 2025, https://docs.python.org/3/library/unittest.html

[20] Pytest [Documentation] (n.d.) Get Started, Accessed March 10, 2025, https://docs.pytest.org/en/stable/getting-started.html

[21] Flask [Documentation] (n.d.) Testing Flask Applications, Accessed March 10, 2025, https://flask.palletsprojects.com/en/stable/testing/

		Cons - Requires installation - Not as versatile as generic Python testing **Applicable** - Yes
	Flask debugger ('Ultra Pythonic', 2025)[22]	Interactive debugger for Flask applications **Pros** - Immediate feedback on errors - Very intuitive - Allows inspecting variables and executing code **Cons** - Cannot be used in production - Can expose sensitive data **Applicable** - Yes
	Python debugger	Python's standard interactive debugger **Pros** - No extra installation - Allows for stepping through code line by line - Useful for inspecting variables and tracing the flow of code execution **Cons** - CLI interface is not user friendly compared to GUI debugging tools. - Manual breakpoint insertion only. **Applicable** - Yes, but not preferable
	VS code debug tools	Integrated debugging features for VS code IDE **Pros** - GUI debugging is MUCH better than CLI debugging - Supports Python debugger - Easy to use **Cons** - Requires setup (often handled by VS Code itself) **Applicable** - Yes!!
	Browser developer tools	Built in tools in web-browsers for debugging, like **Lighthouse.** **Pros**

[22] Ultrapythonic (2025) Flask Basics: Structure, Server Setup, and Debugging Explained, Accessed March 10, 2025, https://ultrapythonic.com/understanding-flask-server-configuration-and-debugger/

		- Essential – allows proper inspection of network traffic and DOM elements - Easy to use and shows everything that the client receives **Cons** - Limited back-end debugging (which is a good thing to be honest) **Applicable** - Yes!
Installation tools	venv / virtualenv (virtual environment)	**Pros** - Isolates project dependencies to prevent conflicts between different projects - Community standard **Cons** - Not intuitive to use **Applicable** - Yes
	pip	**Pros** - Installs, upgrades, and manages python packages - Very easy and intuitive to use - Community standard **Cons** - When something goes wrong pip is not great at telling what the problem is. **Applicable** - Yes
Maintenance	Flask monitoring dashboard	**Pros** - Tracks metrics of Flask applications like response time, helping identify bottlenecks and slow endpoints. - Tracks the execution of path requests **Cons** - Can be a bit heavy to run **Applicable** - Yes
	Prometheus ('Nana', 2020)[23]	**Pros** - Collects and stores data across the whole application, helps to monitor resource usage and application health **Cons**

[23] 'Nana' (2020) How Prometheus Monitoring Works | Prometheus Architecture Explained, Accessed March 10, 2025, https://www.youtube.com/watch?v=h4Sl21AKiDg

		- Can be heavy to run and complex. This is typically for applications that use containers. **Applicable** - No, because my application would not be complex enough to warrant this.
	Sentry ('Sentry', n.d.)[24]	**Pros** - Captures and aggregates errors, identifying errors quickly. - Very good for debugging errors in production. **Cons** - Error tracking (primarily) and not performance monitoring. **Applicable** - Yes, but it might be overkill
	Git version control	**Pros** - Tracks changes to code and files overtime. - Allows rollback to previous versions **Cons** - Does not monitor live systems **Applicable** - Yes. I will be using GitHub in my project anyway.

[24] Sentry (n.d.) Debugging Microservices and Distributed Systems, Accessed March 10, 2025, https://sentry.io/resources/debugging-microservices-and-distributed-systems/

Software Applications

Application type/category	Examples	Pros/cons
Web app (Choice)	Frontend: - React - Angular - Vue.js Backend: - Express.js - Django - Flask - Ruby on rails - .NET Database: - MySQL - PostgreSQL - MongoDB Deploy on: - AWS - Azure - GCP	**Pros** - This was something I have done in my last assessment - If I was to deploy on a platform like GCP, I wouldn't have to worry about creating a whole new server to deploy on. - Online storage **Cons** - I need to worry about security, and as a not so professional developer, one small mistake can cost a lot. Especially because my client actually wants to use the tool to his own benefit. - I might be stuck with a single third-party cloud provider. - Requires internet connection to sync.
Mobile app (Native)	Android: - Kotlin - Java iOS: - Swift	**Pros** - Mobile development is fun and something that I'm not used to, especially designing for mobile apps. - The app itself can enjoy a faster and more robust platform - Can run offline **Cons** - Learning curve!! - Because of the simple nature of work, a mobile only app is just not going to work. - Not cross platform between devices either.
Mobile app (Cross-platform)	- React native (JS) - Flutter (Dart) - Xamarin (C#)	**Pros** - Single codebase that works on multiple platforms. - Relatively alright performance. - Fun challenge too - When designing for cross platform I am not inherently expected to conform to an

		established identity (MUI3 for Android and HIG for iOS), I can establish my own identity. - Can run offline **Cons** - Debugging would be a nightmare. Some platforms might not work well with things that work on other platforms, and this is mobile too which makes it harder. - Another massive learning curve. - Again, when you think of project management applications, you do not really consider mobile.
Desktop app (Native)		**Pros** - Really fast - OS Specific features - Can run offline **Cons** - VERY large learning curve - Not cross platform. - People often look down on native apps unless they're built really well. - I'm building on MacOS, but marking happens on Windows, which means that marking may not even work.
Desktop app (Electron)	Electron	**Pros** - Cross platform!! - Uses HTML CSS and JS which I am used to - Development shouldn't be too hard - Can run offline **Cons** - I can't apply much of my SEN learning.

Chosen: **Flask backend, raw HTML/CSS/JS front end.**

Why Flask?

Flask is based on Python. While express is closer to real world web development, the HSC exam will expect me to provide implementations in Python, not JavaScript.

Using Flask will allow me to sharpen my Python skills, like OOP, while doing my major project.

Why not Electron?

The point of using Electron was because it removed the online parts and was just something that you could install. This way, I did not have to worry about cybersecurity, however as cyber is a large part of the HSC course for Software Engineering, building a project that has no cybersecurity concerns backfires.

Instead, I will still implement cybersecurity, but instead of making it as secure as possible, I will simply focus on learning how to ensure that a sufficient degree of security exists.

Why not Express.js?

While Express.js is industry standard, JavaScript is not assessed in the HSC exam, and hence I'll stick with Flask. After the HSC, I'll have more freedom to explore industry standards to a greater extent.

Implementation method

Students **explain** the applicability of the implementation method for the current project. These are normally: direct, phased, parallel and pilot.

Implementation methods refer to methods that allow a product to transition from their most mature and stable development branches to production[25]. Different approaches to this have various pros/cons[26].

Direct	Direct implementation means to directly cut from development to production. There is no transition, and the deployed system has instantly changed. Because there is no other service required to host old systems or there is no need to dissect the existing branch into parts, it's the easiest and cost-effective deployment solution. However direct implementation can be risky, as a single bug is experienced by the entire userbase. If a bug is critical, the entire

[25] Computersciencewiki (2023) Implementation methods, Accessed May 2 2025, https://computersciencewiki.org/index.php/Implementation_methods

[26] universalCPAreview (n.d.) What are the different types of approaches to implement a new system?, Accessed May 2, 2025, https://www.universalcpareview.com/ask-joey/what-are-the-different-types-of-approaches-to-implement-a-new-system/

	userbase experiences it and the system may be down. Direct implementation is also more susceptible to the exploitation of new vulnerabilities.
Parallel	Parallel implementation refers to when the old system and run in parallel with the new system. Often, this can be seen when there is an option to switch from the old system to the new system to the end user. This method allows users to provide sufficient feedback to the new system so when put into production, the new system fulfills user needs as best as it can. However parallel requires double the existing number of resources to facilitate the two systems, which can be economically infeasible.
Phase-in	Phase-in means that the system is slowly being transitioned to the new system by breaking the new system into parts. This slow and progressive rollout system allows the intake of feedback to be processed in a more effective, targeted, and responsive approach. While low-risk, this method requires significantly more time and effort to go from idea to reality. In a team environment, groups would be established with clear roles to split apart the proposed new system and to collect/implement feedback accordingly.
Pilot	A pilot approach refers to when the new product is created in a dedicated environment with assigned testers to properly identify and fix bugs. Piloted approaches allow issues that are found in the new system

	to be identified and fixed in a time effective manner, while also taking feedback and implementing that feedback in a non-destructive way. While pilot approaches are relatively easy to setup, gaining dedicated testers to enter the pilot program can become problematic. Furthermore, the testers may not be entirely representative to the entire demographic of the intended userbase, potentially skewing feedback and judgement.

All of these approaches refer to implementation strategies to deploy a product. Pilot is great for applications with a dedicated, enthusiastic tester base. Parallel is great for applications with a large userbase but no dedicated pilot programs. Phase-in is great for applications with a robust developer team.

However, Genta has only one developer and a handful of (almost entirely non-technical) testers. Genta will also not have a final release date with dedicated, active users until its scheduled release which is set for the submission date of this project.

Using this information, I find that it is economically sound and time effective to use the **direct implementation** approach.

The significant issue to this approach is risk. To mitigate this:

- I will set up and conduct thorough internal testing to avoid critical bugs and vulnerabilities in production.
- I will also aim to conduct focused feedback sessions with testers with their specific scenarios to identify usability issues and errors.
- I will also set up monitoring and alerting systems, such as those available on AWS, to avoid significant errors/vulnerabilities, preserving the confidentiality, integrity, and availability of user data, along with avoiding cost hikes.

Jump to Table of Contents

Problem context

Generate requirements including functionality and performance.

My application is a PWA (progressive web app), which is an application that runs inside a browser on the web. The nature of progressive web apps ensures an expectation for the experience expected by the application. These expectations influence the requirements for the functionality and performance of my webapp.

Hence, here are Genta's functionality and performance requirements:

- As PWAs are built on the web, Genta will use HTML, CSS, and JS and be coded for maximum compatibility. This means using libraries that are compatible with low-end devices or designing with responsiveness as a priority.
- A PWA should mimic an app and not feel like a website. Therefore, their functionality is not an afterthought, it is integrated by design. This can be implemented in Genta through background sync and a smooth, app-like UI.
- Every device has a web browser, but not every device is created equal - hence PWAs should progressively scale their functionality based on their platform. This can come in the form of:
    - If a browser supports it, adding push notifications to when events are close to coming up
    - If a browser supports it, ensuring Genta is installable
    - If a browser supports it, use service workers to enable offline functionality
- End users find the UI of well-designed, app-like experiences secure. Genta needs to be able to meet that expectation and needs to be a secure application.

Quality assurance, quality criteria, compliance, and legislative requirements

Investigate how software engineering solutions are quality assured, including:

- Defining criteria on which quality will be judged
- Ensuring requirements are met using a continual checking process
- Addressing compliance and legislative requirements

Quality assurance and quality criteria

According to asq.org (n.d.)[27], the terms quality assurance (QA) and quality control (QC) are interrelated, but have separate meanings. Quality assurance is closer to how a quality system is designed to assure a team that the final product is of sufficient quality, and quality control is a subset of that, where it is more of the activities and techniques that allow a team to fulfill quality requirements. Mudali (n.d.)[28] shows that good QA means that not only a framework is created, but that the framework is met, and the project is continuously meeting those requirements. Through this, the project requires less reworks, fewer springs, and the outcomes of the project meet its initial goals proactively.

Quality criteria	Explanation
Usability	The user experience for Genta needs to be intuitive and easy to work with in order for efficient navigation. Users need to be able to set their projects, create new events/tasks/milestones, and to be able to edit them accordingly with a minimal number of clicks/taps.
Functionality	Corresponding to Genta's MVP, Genta needs to be able to provide a platform that reliably creates, edits, and saves projects, events, and the data within them.
Performance	Genta should respond quickly and smoothly to user interactions even when internet connection speeds are slow. This includes the smooth navigation of tasks and projects, on both desktop and mobile.
Data integrity	User data, including projects and events, should be stored securely and retrieved accurately and reliability. Data needs to be sent from

[27] ASQ.org (n.d.) Quality Assurance vs Control, Accessed March 7 2025, https://asq.org/quality-resources/quality-assurance-vs-control

[28] Mudal (n.d.) the fundamentals of quality assurance (QA) and it's impact on organisational performance, Accessed March 7 2025, https://conqa.com/qafundamentals

	the database to the client securely, through methods like HTTPS encryption in transit.
Security	Sign in with Google needs to be implemented according to Google's official, latest guidelines, ensuring that the IdP is implemented securely and reliably.
Boundaries	Genta needs to adhere to the originally set boundaries, like no AI, and no collaboration. This ensures that Genta remains focused to my client's original goal.

Legal compliance and legislative requirements

Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian NSW Privacy and Personal Information Act (1998) and/or Federal Privacy Act (1988). Alternatively, international standards on information security management such as ISO/IEC 27001 may also be applicable.

Compliance or legislative issue	Methods for mitigation
Handling of personal information	Immediate personal data is not explicitly required for Genta, as Google Identity handles otherwise required sensitive information for authentication on their services. This means that the sensitive information is currently under the legal liability of Google, and not Genta. To mitigate, I'll continue developing Genta around Sign in with Google, ensuring that minimal data outside of that is collected. I will also avoid building features that may require me to access more sensitive data.
User input data	While Genta does not explicitly request for sensitive information, users may choose to input potentially sensitive information onto areas like the to-do list or notes section. In the event of a data breach, this potentially sensitive information may be leaked to the public. To mitigate this, I'll add disclaimers which notify individuals that they should not enter any dangerously sensitive information into Genta. I'll use systems like encryption at rest, strong database

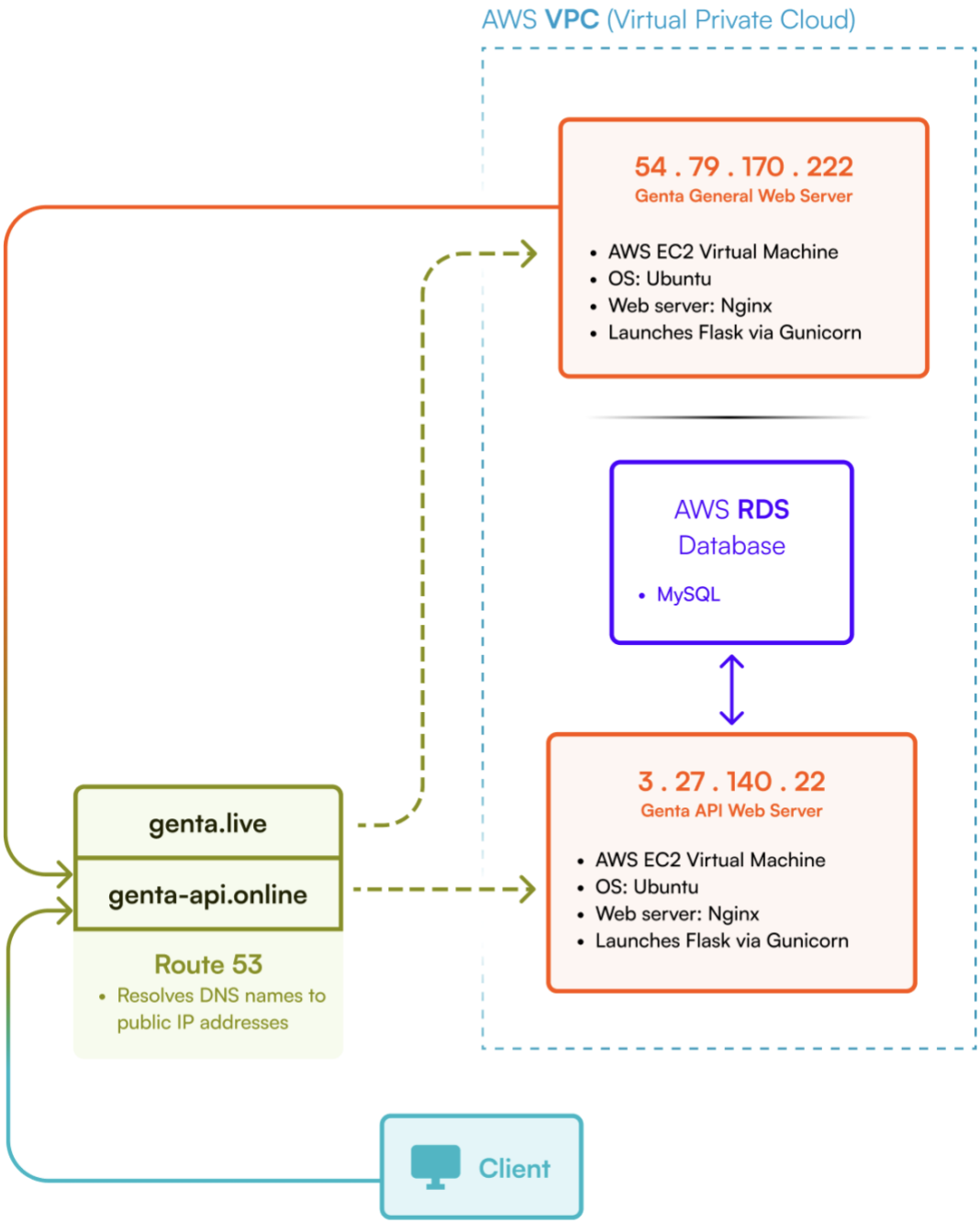
	passwords, database connection access limited to only the virtual machine and nothing else (no 0.0.0.0/0 access). In terms of a data breach, if the breach satisfies the requirements of the Notifiable Data Breach (NDB) scheme, I must report the data breach to the OAIC[29].
General data security	Transmission, handling, and storage of data between the backend and the frontend (along with steps outside of those) can pose security vulnerabilities that may compromise the confidentiality, availability, and integrity of the data. To mitigate this, I'll use secure development practices, like setting up appropriate CSPs, CORS policies, using deployment-focused servers in production and using secure communication protocols like HTTPS.

Finalised implementation

I'm using Amazon VPC (Virtual Private Cloud), containing 2x (1x backend and 1x frontend) Amazon EC2 (Elastic Cloud Compute) Ubuntu virtual servers which run Flask via Gunicorn behind Nginx and 1x Amazon RDS (Relational Database Service) as MySQL.

To access the EC2 instances, I use Amazon Route 53 which is a global DNS service, routing https://genta.live and https://genta-api.online to their respective EC2 instances. During usage, users will access https://genta.live which fetches from the backend api https://genta-api.online.

[29] OAIC (n.d.) When to report a data breach, Accessed 3 May, 2025, https://www.oaic.gov.au/privacy/notifiable-data-breaches/when-to-report-a-data-breach



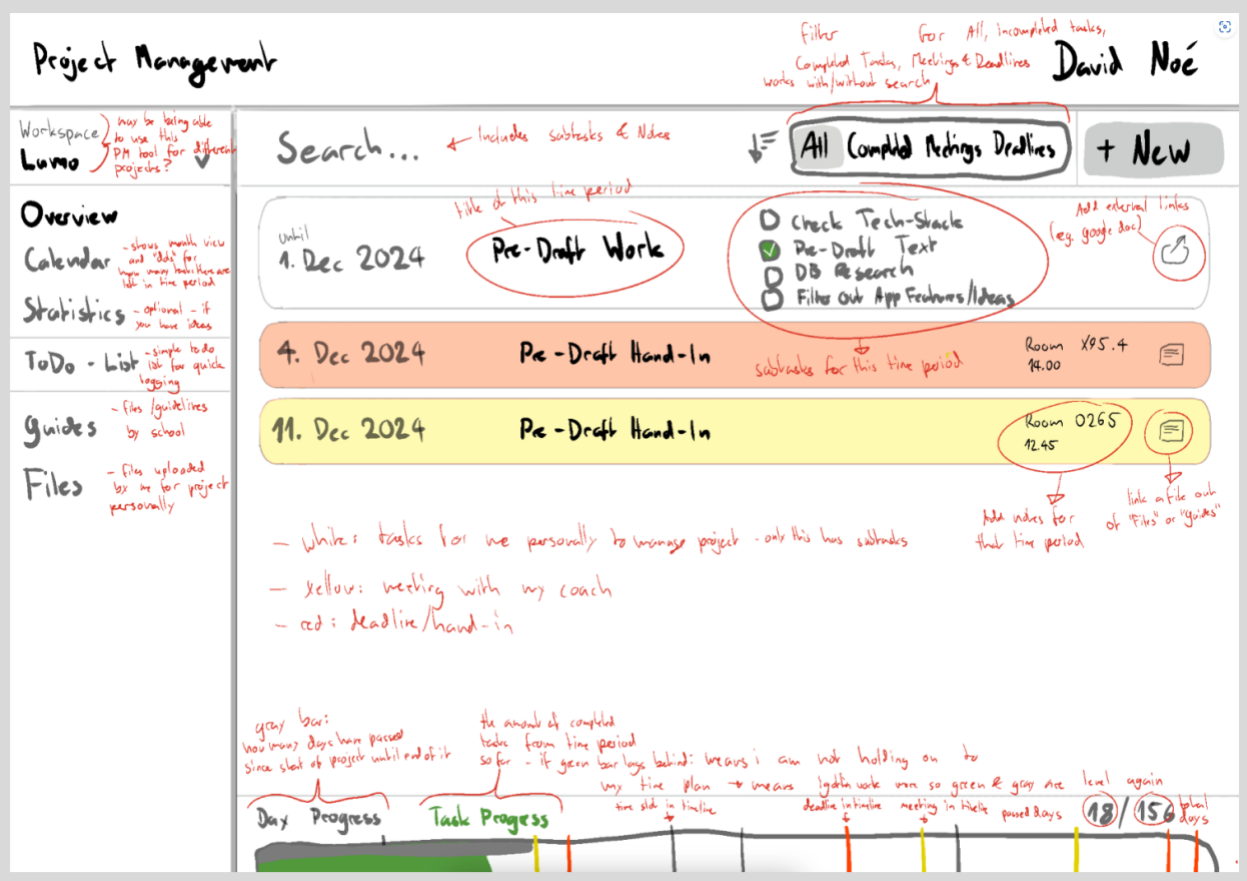
NOTE: VPC -> The VPC has 3 subnets across three availability zones around Asia-Pacific, so, realistically, segments of this diagram would be duplicated 3 times. This is not shown because it is not useful information.

Systems modelling

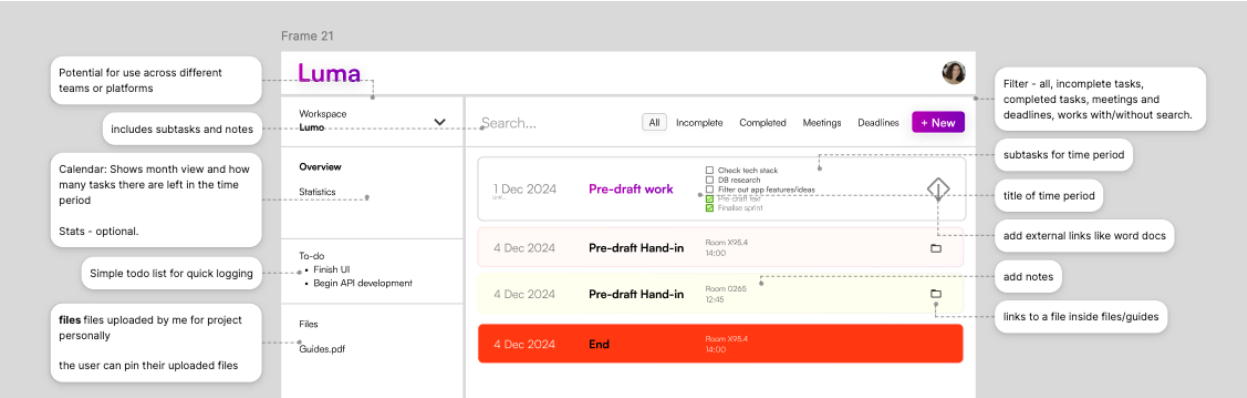
Demonstrate the use of modelling tools.

UI Design + **Storyboard**

My client, David, kicked off the UI design process by sending this drawing of what he wanted Luma to look like.



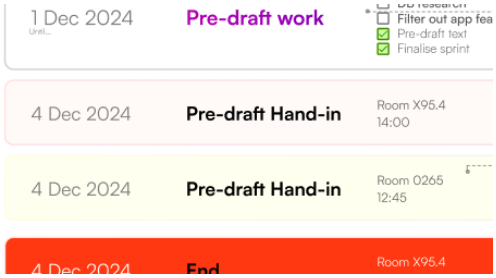
From that, I mocked a quick UI



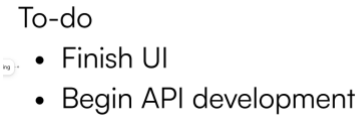
David and I had a meeting to discuss the semantics further. We found that:

- Users should have the ability to create new workspaces, logically separating projects apart

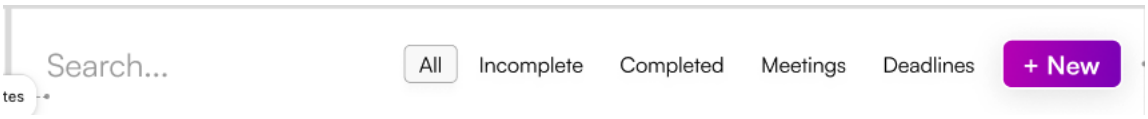
- For the main section, users will have numerous cards representing a block that needs to be completed. Inside each card is either a to do list and/or notes.



- Users could add files/links to each card
- On the side, there could be a whole workspace to do list

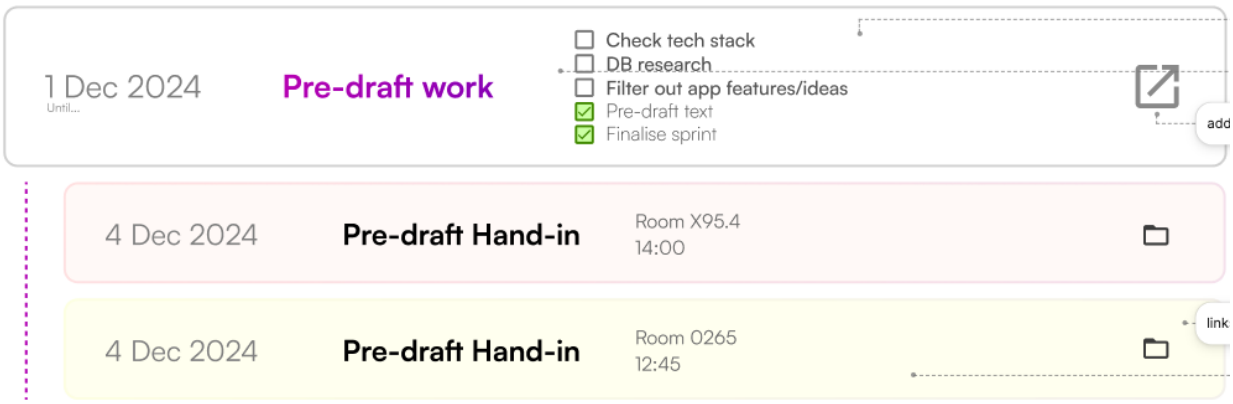


- A search bar could make cards easier to find.



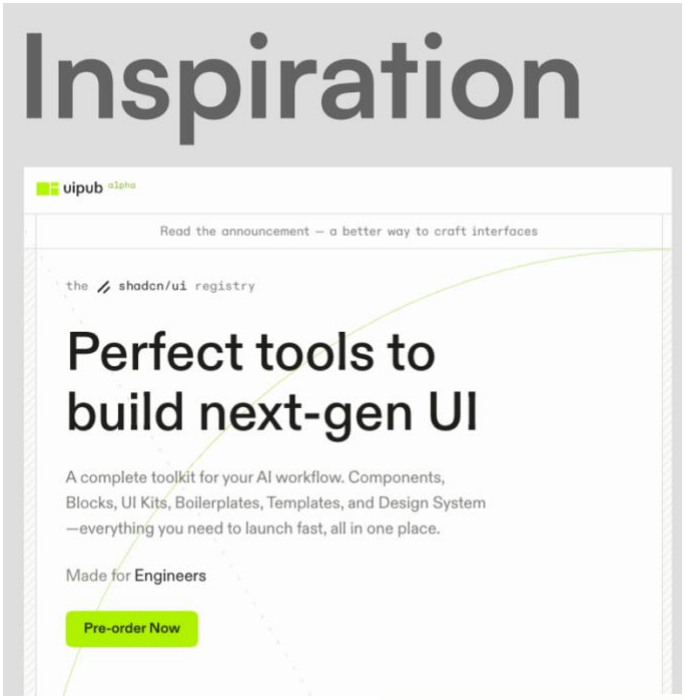
In another meeting, we refined the idea for cards, where a card could have sub tasks.

Jump to Table of Contents

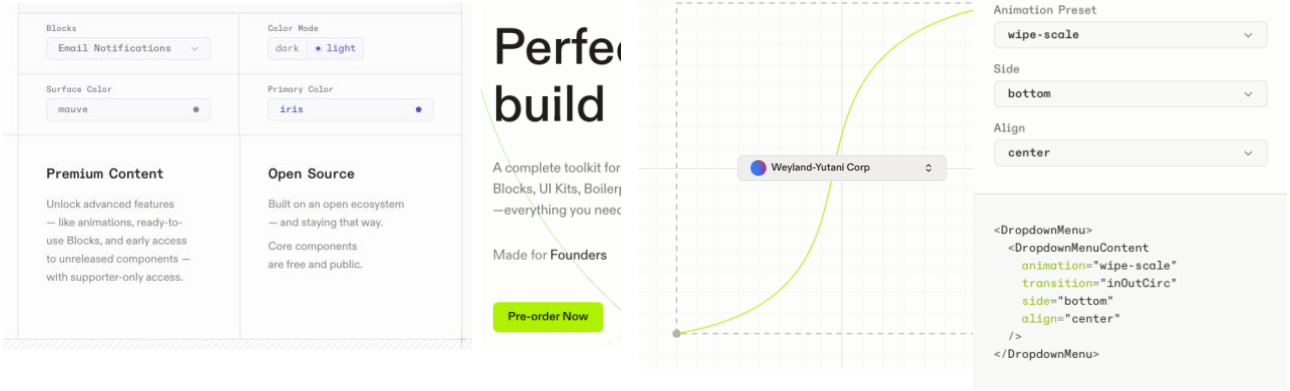


Now, I needed to significantly clean up the design, making it responsive and closer to a final product. However, to me the current design still does not make sense semantically, and is overly complex in some areas. It also looks far too generic.

Hence, I took inspiration from https://uipub.com/, which you can see on the side.


Inspiration
uipub alpha
Read the announcement — a better way to craft interfaces
the shadcn/ui registry
Perfect tools to build next-gen UI
A complete toolkit for your AI workflow. Components, Blocks, UI Kits, Boilerplates, Templates, and Design System —everything you need to launch fast, all in one place.
Made for Engineers
Pre-order Now

Uipub uses a very geometric design language with soft general colours contrasting with a brutalist green. This makes for an appeasing UI style.



Using these principles, I created this first draft of the UI. I also renamed Luma to Genta

The UI design displayed shows multiple components that try to work together to form one coherent system.

The large “Today” section orients Genta into tool that offers chronological workflow.

A single card, “Software Engineering Major” acts as a project/card (need to fix later).

The card shows a to do section, where users can add items to their checklist, and for each item they can add further descriptions. They can check/uncheck each item.

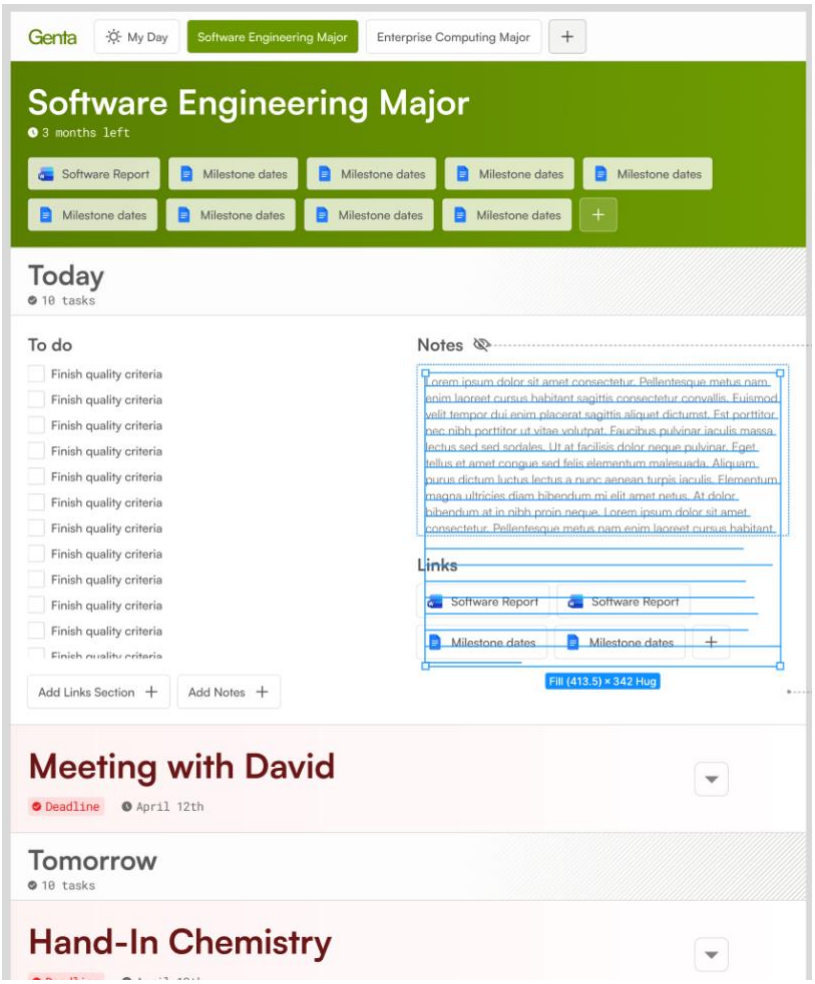
The links section shows various links that are related to the card. The notes section allows the user to add any text in a format that does not conform to any existing structure.

After a meeting, David and I found the following problems with this design:

- How do we delete to do items?
- How do we delete links?
- What happens when there are no todo items/links/notes? Can we disable the sections?
- How do we delete a card?
- Does the card represent a single project or an event?
- Is the “Today” necessary.
- What happens if the items overflow?

Jump to Table of Contents

David and I worked through these problems and created the design on the side.



Projects are now logically separated with a top navigation bar. This solves the problem of "does the card represent a single project or event?", as now a card represents an event.

A hero section now shows key details about the project, where the project can have links associated with it on the actual project. A green background makes it very clear.

If the to-do or notes section overflows, the inner content within the div will just be scrollable after it hits a maximum height. This is shown by the cut of items in the todo section, and the highlighted notes inside the notes section which are hidden at the overflow.

Different cards show "meeting with David" and "hand in chem" which will have their own todo and notes section.

There are still problems with this design:

- How to add new projects?
- How to add a new deadline?

Jump to Table of Contents

In the third and final major iteration of this design, David and I made numerous changes to push the design as close as development ready as possible

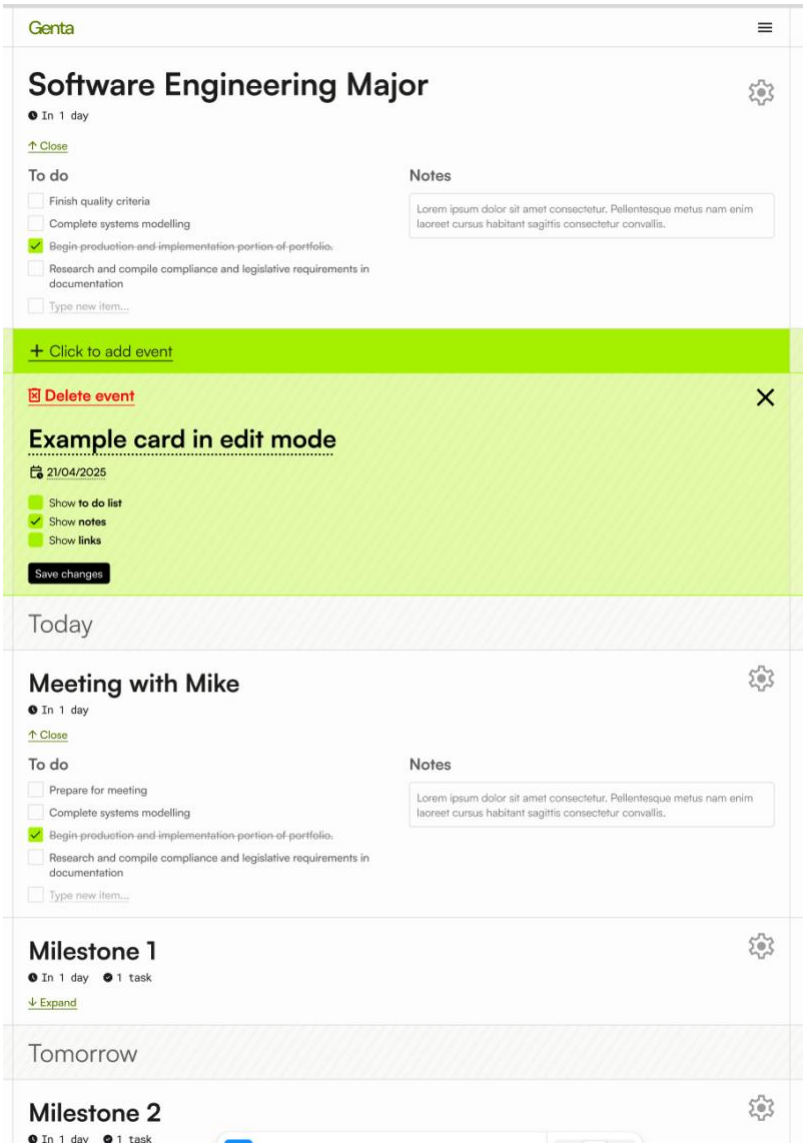
All projects are now stored in a hamburger menu on the top right corner. This makes sense for mobile users too.

The project – Software Engineering Major – is shown as an example. Projects themselves now have their own to do and notes section. To make development easier, links have been removed.

Each card has a cog, and when clicked, it opens edit mode. Edit mode is also opened when a new event is added, which can be done using the Click to add event bar. Projects have their own edit mode, shown below.

Within edit mode, the user can edit the title, date, and turn on/off the notes and todo sections. This shouldn't delete any existing data.

Events can be collapsed to save space.



Each event is ordered by how long away it is.

Jump to Table of Contents

JSON outline

Genta-api.online will send data to genta.live in the form of a JSON. To help me in my system modelling, I've decided to start with a concept of what this JSON will look like. First shown is the concept JSON in its entirety, and then there is a breakdown and explanation of each part.

```
[
  {
    "id":0,
    "projectTitle":"Project Title",
    "dueDate":"2025-8-15",
    "events":[
      {
        "id": 0,
        "collapsed":true,
        "title":null,
        "dueDate":null,
        "todo":[
          {
            "id":0,
            "checked":false,
            "content":"This item is uncompleted"
          },
          {
            "id":1,
            "checked":true,
            "content":"This item is completed!"
          }
        ],
        "notes":"Text inside notes"
      },
      {
        "id": 1,
        "collapsed":true,
        "title":"Event 2",
        "dueDate":"2025-7-13",
        "todo":null
        "notes":"This is an eaxmple of the notes section!"
```

```
      }
    ]
  }
]
```

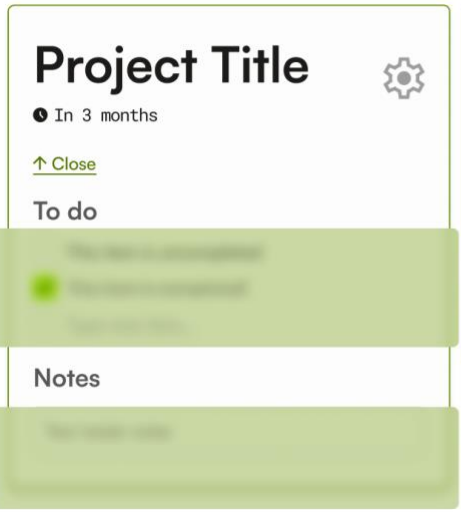
UI Element

For each project, there is a **project title** and a **due date.**
In the SQL, each project will be assigned a primary key shown as the **ID.**
Every project also has events

Demo JSON

```
[
    {
        "id":0,
        "projectTitle":"Project Title",
        "dueDate":"2025-8-15",
    },
    {
```

UI Element



Demo JSON

```
"id":0,
"projectTitle":"Project Title",
"dueDate":"2025-8-15",
"events":[
    {
        "id": 0,
        "collapsed":false,
        "title":null,
        "dueDate":null,
        "todo":[…
        ],
        "notes":"
    },
```

In the project, the user may want to add a todo list or notes section in the header. This is sent as **the first event.**

Jump to Table of Contents

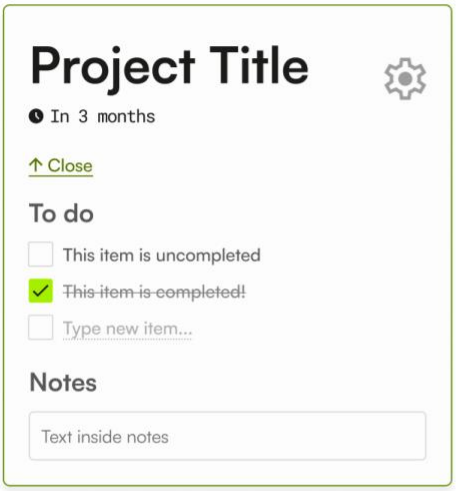
UI Element

Users can choose to collapse/expand their todo/notes section. This is shown using the **collapsed** tag.

Demo JSON

```
{
    "id":0,
    "projectTitle":"Project Title",
    "dueDate":"2025-8-15",
    "events":[
        {
            "id": 0,
            "collapsed":true,
            "title":null,
            "dueDate":null,
            "todo":[…
            ],
            "notes":"This is an eaxmple of the notes section!"
```

UI Element

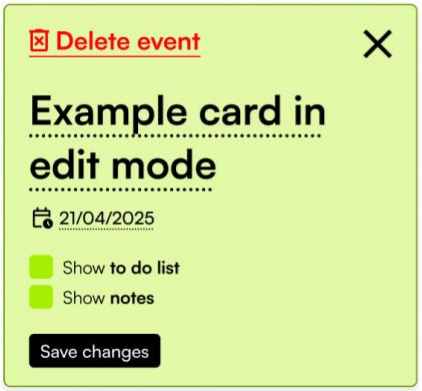


Notes and todos are stored inside the events.
To do items have a state of being either checked or unchecked.

Demo JSON

```
"events":[
    {
        "title":null,
        "dueDate":null,
        "todo":[
            {
                "id":0,
                "checked":false,
                "content":"This item is uncompleted"
            },
            {
                "id":1,
                "checked":true,
                "content":"This item is completed!"
            }
        ],
        "notes":"Text inside notes"
    },
```

UI Element

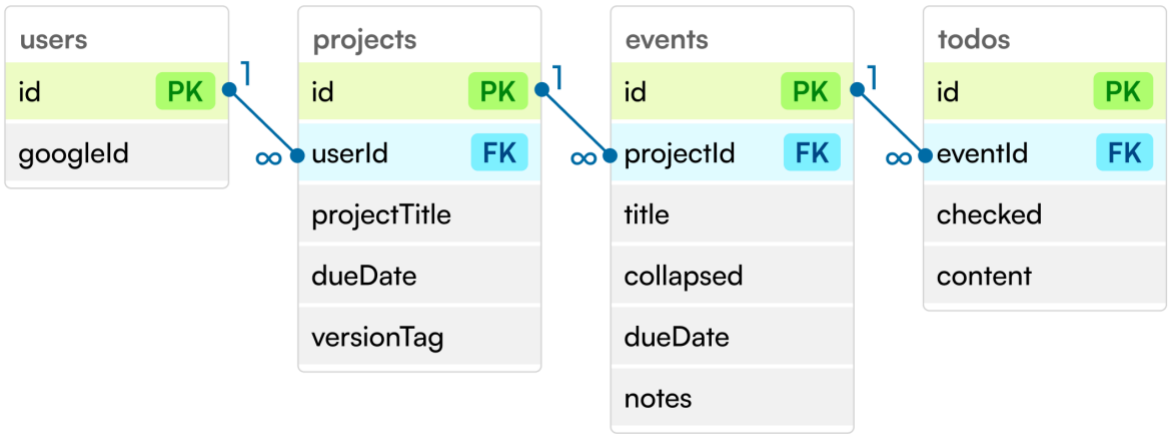


Demo JSON

```
"todo":null,
"notes":null
```

When notes or to do sections are hidden, they appear is “null” in the JSON.

Schema



Database data dictionary

Using the schema, I can create a database dictionary. I learnt how to use constraints from w3schools[30].

Table	Column	Data type	Constraints	Example	Description
users	**id**	**INTEGER**	**PRIMARY KEY, AUTO_INCREMENT**	**1, 2, 3**	**Identifier for users in database**
	googleId	VARCHAR(21)	NOT NULL	a3k4jf...	Google identifier from Google API Given by the SUB attribute in returned JSON
	versionTag	VARCHAR(255)	NOT NULL	Klasdf	ID to help prevent stale overwrites
projects	**id**	**INTEGER**	**PRIMARY KEY, AUTO_INCREMENT**	**1, 2, 3**	**Identifier for projects**
	userId ↗ users.id	**INTEGER**	**FOREIGN KEY, NOT NULL**	**1, 2, 3**	**Identifies who the**

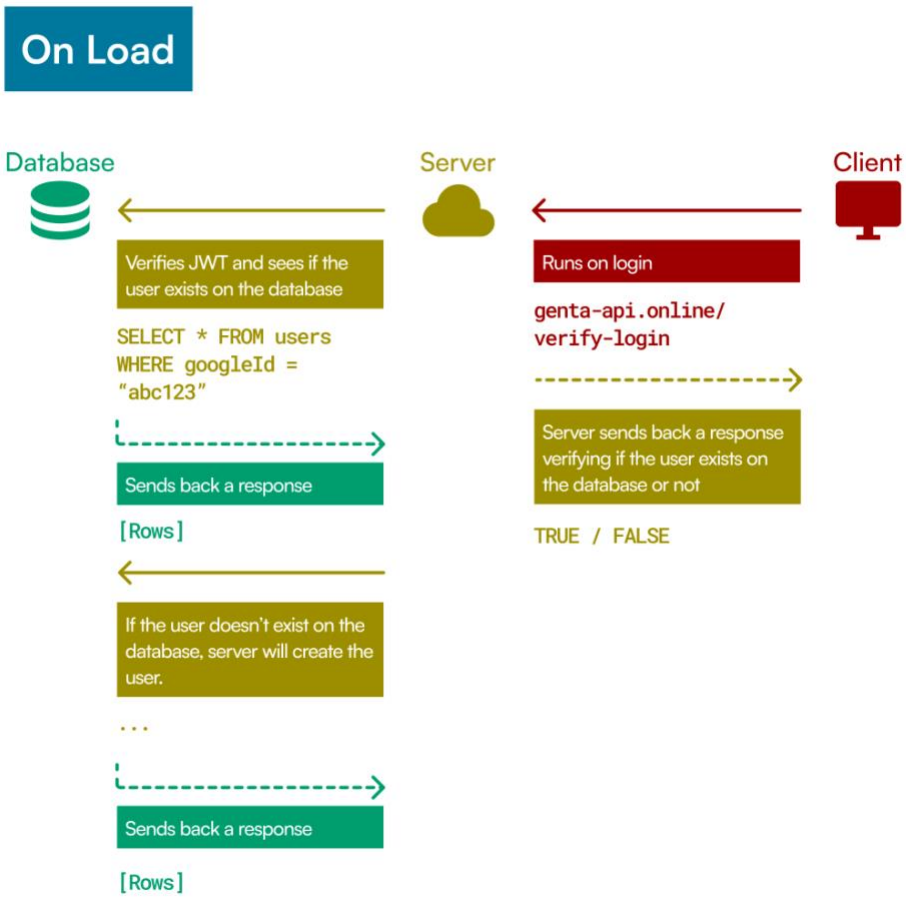
[30] W3Schools (n.d.) My SQL Constraints, Accessed 3 May 2025, https://www.w3schools.com/mysql/mysql_constraints.asp

					project belongs to
	projectTitle	VARCHAR(255)	NOT NULL	“SEN Major”	The title of the project
	dueDate	DATE	NOT NULL	2025-07-12	When the project is due
events	**id**	**INTEGER**	**PRIMARY KEY, AUTO_INCREMENT**	**1, 2, 3**	**Identifier for events**
	projectId ↗ projects.id	**INTEGER**	**FOREIGN KEY, NOT NULL**	**1, 2, 3**	**Identifies what project the event belongs to**
	title	VARCHAR(255)		“Meeting with Matt”	The title of the event
	collapsed	BOOLEAN	NOT NULL, DEFAULT FALSE	TRUE, FALSE	If the notes/todo section is collapsed for the event or not
	dueDate	DATE	NOT NULL	2025-08-15	When the event is due
	notes	TEXT		“This is my notes section”	The content inside the notes section
	todoShown	BOOLEAN	NOT NULL, DEFAULT TRUE	TRUE, FALSE	Hide/show todo section
	notesShown	BOOLEAN	NOT NULL, DEFAULT TRUE	TRUE, FALSE	Hide/show notes section
todos	**id**	**INTEGER**	**PRIMARY KEY, AUTO_INCREMENT**	**1, 2, 3**	**Identifier for todo items**
	eventId ↗ events.id	**INTEGER**	**FOREIGN KEY, NOT NULL**	**1, 2, 3**	**Identifies what event the todo belongs to**

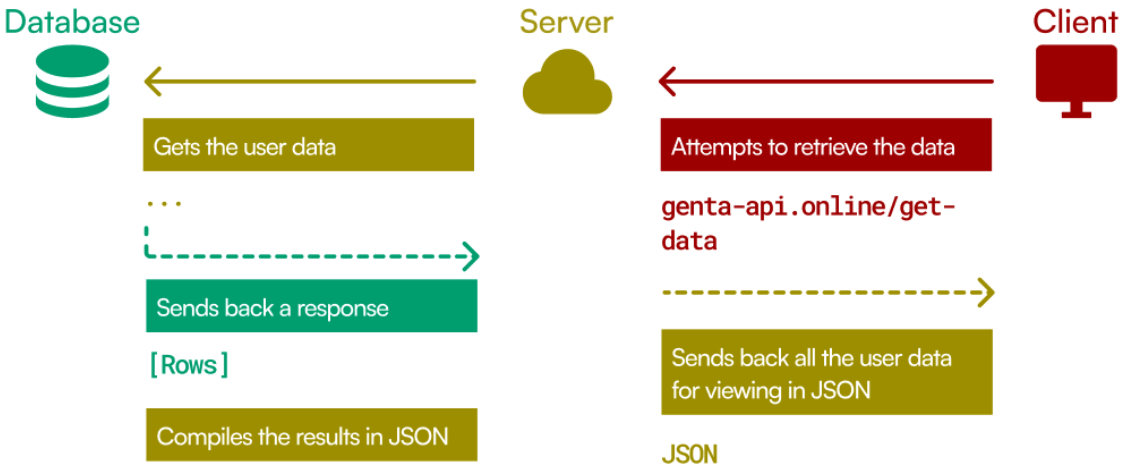
	checked	BOOLEAN	NOT NULL, DEFAULT FALSE	TRUE, FALSE	If the item is checked off or not.
	content	TEXT		“Go to bed”	The content inside the todo item.

API routes and general server handling.

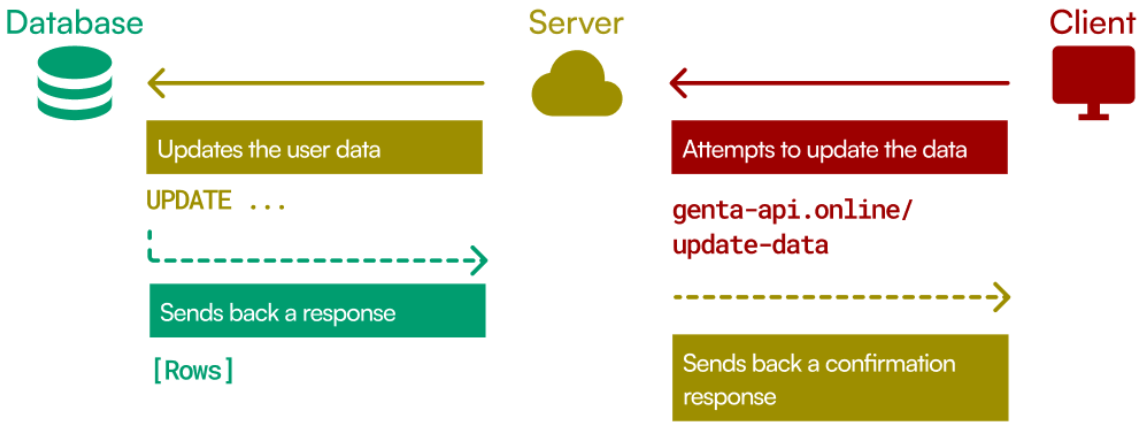
From here I can figure out how the API routes will interact with the database and how the frontend will communicate with the backend for creating and updating.


On Load
Database
Server
Client
Verifies JWT and sees if the user exists on the database
SELECT * FROM users WHERE googleId = “abc123”
Sends back a response
[Rows]
Runs on login
genta-api.online/ verify-login
Server sends back a response verifying if the user exists on the database or not
TRUE / FALSE
If the user doesn’t exist on the database, server will create the user.
...
Sends back a response
[Rows]

When loading user data



When updating user data



Jump to Table of Contents

Pseudocode

Notes:

- The frontend needs to store a version ID.
    - This is cleared whenever the frontend successfully retrieves from `genta-api.online/get-data`.
    - If the frontend cannot find a valid version in local storage it needs to run get-data (or whenever reloaded_
    - Version ID is sent with every UPDATE request to genta-api.online/update-data and is cross-checked on the server side
        - If it matches with the last updated id on the database, then the request updates anything on the server. Then the server responds with a new version ID which is stored on localstorage on the client side
        - If it doesn't match with the last updated id on the database, the server responds with a 400 response.
- When creating a new project/event/todo, the client adds a corresponding object to the JSON

https://genta-api.online> server pseudocode

```
Import necessary packages
Set sensitive variables with their environment variables.
Connect to the database
END ()

->>>> Begin vendor code <<<<.
BEGIN verify_google_token(token)
    Where token is the JWT (JSON Web Token), vendor provided code
    will verify if the JWT has been issued correctly.
END verify_google_token(token)

BEGIN token_required(f)
    Vendor provided code checks if a token exists in the authorisation
    header, then runs verify_google_token to verify it.
END token_required(f)
->>>> End vendor code <<<<.

-> API Route.
```

```
When routed to "/verify-login"...
Check the authorisation header using the token_required function
BEGIN home ()
    response = HTTP 200 response
    -> This means that the token_required function has passed
    -> and the user has successfully signed in with google.
    RETURN response
END home ()

-> Get projects and user details into a structured object.
BEGIN getUserDataFromDB (user_google_id)
    user_record = SQL Find row in table {users} that has {googleId} = user_google_id
    user_db_id = NONE
    user_version_tag = NONE

    IF user_record = none THEN
        new_version_tag = generate a new version tag
        user_db_id = SQL Create new user in table {users} with ({googleId} = user_google_id, {versionTag} = new_version_tag)
RETURNING id
        user_version_tag = new_version_tag
    ELSE
        user_db_id = user_record.id
        user_version_tag = user_record.versionTag
    ENDIF

    projects_from_db = SQL Find all rows in table {projects} that have {userId} = user_db_id
    assembled_projects_data = []

    IF there are items inside projects_from_db THEN
        FOR i = 0 TO length of projects_from_db STEP 1
            current_project_from_db = projects_from_db[i]
            project_dict = {
                "id": current_project_from_db.id,
                "projectTitle": current_project_from_db.projectTitle,
                "dueDate": current_project_from_db.dueDate,
                "events": []
            }

            events_from_db = SQL Find all rows in table {events} that have {projectId} = current_project_from_db.id
```

```
        IF there are items inside events_from_db THEN
            FOR j = 0 TO length of events_from_db STEP 1
                current_event_from_db = events_from_db[j]
                event_dict = {
                    "id": current_event_from_db.id,
                    "title": current_event_from_db.title,
                    "collapsed": current_event_from_db.collapsed,
                    "dueDate": current_event_from_db.dueDate,
                    "notes": current_event_from_db.notes,
                                "notesShown": current_event_from_db.notesShown,
                                "todoShown": current_event_from_db.todoShown,
                    "todo": []
                }

                todos_from_db = SQL Find all rows in table {todos} that have {eventId} = current_event_from_db.id
                IF there are items inside todos_from_db THEN
                    FOR k = 0 TO length of todos_from_db STEP 1
                        current_todo_from_db = todos_from_db[k]
                        todo_dict = {
                            "id": current_todo_from_db.id,
                            "checked": current_todo_from_db.checked,
                            "content": current_todo_from_db.content
                        }
                        Append todo_dict to event_dict.todo
                    NEXT k
                ENDIF
                Append event_dict to project_dict.events
            NEXT j
        ENDIF
        Append project_dict to assembled_projects_data
    NEXT i
ENDIF

RETURN {
    "user_db_id": user_db_id,
    "user_version_tag": user_version_tag,
    "projects": assembled_projects_data
}
END getUserDataFromDB
```

```
-> API Route for getting user data.
When routed to "/get-data"...
Check the authorisation header using the token_required function
BEGIN getData ()
    user_google_id = Extract the Google ID of the user from the JWT

    user_and_projects_data = getUserDataFromDB(user_google_id)

    IF user_and_projects_data.user_db_id = NONE THEN
        RETURN HTTP 500 with JSON {"error": "User processing failed."}
    ENDIF

    response_data = {
        "user_id": user_and_projects_data.user_db_id,
        "versionTag": user_and_projects_data.user_version_tag,
        "projects": user_and_projects_data.projects
    }
    jsonified_response = convert response_data to JSON
    RETURN jsonified_response
END getData ()


BEGIN editDatabase (user_db_id, projects_from_client_list, existing_projects_list_from_db)

    processed_project_ids_from_client = create an empty set

    FOR project_from_client IN projects_from_client_list
        existing_project_in_db = NONE

        IF project_from_client.id IS NOT NONE THEN
            existing_project_in_db = Find project in existing_projects_list_from_db with id = project_from_client.id
            IF existing_project_in_db IS NONE THEN
                RETURN ERROR ("Project with ID " + project_from_client.id + " to update not found in database for this user.")
            ENDIF

            current_project_id_in_db = existing_project_in_db.id
            IF project_from_client.projectTitle IS NOT EQUAL TO existing_project_in_db.projectTitle THEN
```

```
            SQL Update table {projects} SET {projectTitle} = project_from_client.projectTitle WHERE {id} =
current_project_id_in_db
        ENDIF
        IF project_from_client.dueDate IS NOT EQUAL TO existing_project_in_db.dueDate THEN
            SQL Update table {projects} SET {dueDate} = project_from_client.dueDate WHERE {id} = current_project_id_in_db
        ENDIF

        Add current_project_id_in_db to processed_project_ids_from_client
    ELSE
        new_project_id = SQL Create new row in table {projects} with (
            {userId} = user_db_id,
            {projectTitle} = project_from_client.projectTitle,
            {dueDate} = project_from_client.dueDate,
        ) RETURNING id
        current_project_id_in_db = new_project_id

    ENDIF

    events_from_client = project_from_client.events
    existing_events_in_db_for_project = []
    IF existing_project_in_db IS NOT NONE THEN /ts
        existing_events_in_db_for_project = existing_project_in_db.events
    ENDIF

    processed_event_ids_from_client_for_this_project = create an empty set

    FOR event_from_client IN events_from_client
        existing_event_in_db = NONE
        current_event_id_in_db = NONE

        IF event_from_client.id IS NOT NONE THEN
            existing_event_in_db = Find event in existing_events_in_db_for_project with id = event_from_client.id
            IF existing_event_in_db IS NONE THEN
                RETURN ERROR ("Event with ID " + event_from_client.id + " to update not found under project " +
current_project_id_in_db)
            ENDIF

            current_event_id_in_db = existing_event_in_db.id
            IF event_from_client.title IS NOT EQUAL TO existing_event_in_db.title THEN
```

```
            SQL Update table {events} SET {title} = event_from_client.title WHERE {id} = current_event_id_in_db
        ENDIF
        IF event_from_client.collapsed IS NOT EQUAL TO existing_event_in_db.collapsed THEN
            SQL Update table {events} SET {collapsed} = event_from_client.collapsed WHERE {id} =
current_event_id_in_db
        ENDIF
        IF event_from_client.dueDate IS NOT EQUAL TO existing_event_in_db.dueDate THEN
            SQL Update table {events} SET {dueDate} = event_from_client.dueDate WHERE {id} = current_event_id_in_db
        ENDIF
        IF event_from_client.notes IS NOT EQUAL TO existing_event_in_db.notes THEN
            SQL Update table {events} SET {notes} = event_from_client.notes WHERE {id} = current_event_id_in_db
        ENDIF

         IF event_from_client.todoShown IS NOT EQUAL TO existing_event_in_db.todoShown THEN
            SQL Update table {events} SET {todoShown} = event_from_client.todoShown WHERE {id} =
current_event_id_in_db
         ENDIF
        IF event_from_client.notesShown IS NOT EQUAL TO existing_event_in_db.notesShown THEN
            SQL Update table {events} SET {notesShown} = event_from_client.notesShown WHERE {id} =
current_event_id_in_db
         ENDIF

        Add current_event_id_in_db to processed_event_ids_from_client_for_this_project
    ELSE
        new_event_id = SQL Create new row in table {events} with (
            {projectId} = current_project_id_in_db,
            {title} = event_from_client.title,
            {collapsed} = event_from_client.collapsed,
            {dueDate} = event_from_client.dueDate,
            {notes} = event_from_client.notes
                    {todoShown} = event_from_client.todoShown,
            {notesShown} = event_from_client.notesShown

        ) RETURNING id
        current_event_id_in_db = new_event_id
    ENDIF

    todos_from_client = event_from_client.todo
    existing_todos_in_db_for_event = []
```

```
        IF existing_event_in_db IS NOT NONE THEN
            existing_todos_in_db_for_event = existing_event_in_db.todo
        ENDIF

        processed_todo_ids_from_client_for_this_event = create an empty set

        FOR todo_from_client IN todos_from_client
            existing_todo_in_db = NONE

            IF todo_from_client.id IS NOT NONE THEN
                existing_todo_in_db = Find todo in existing_todos_in_db_for_event with id = todo_from_client.id
                IF existing_todo_in_db IS NONE THEN
                    RETURN ERROR ("Todo with ID " + todo_from_client.id + " to update not found under event " +
current_event_id_in_db)
                ENDIF

                current_todo_id_in_db = existing_todo_in_db.id
                IF todo_from_client.checked IS NOT EQUAL TO existing_todo_in_db.checked THEN
                    SQL Update table {todos} SET {checked} = todo_from_client.checked WHERE {id} = current_todo_id_in_db
                ENDIF
                IF todo_from_client.content IS NOT EQUAL TO existing_todo_in_db.content THEN
                    SQL Update table {todos} SET {content} = todo_from_client.content WHERE {id} = current_todo_id_in_db
                ENDIF
                Add current_todo_id_in_db to processed_todo_ids_from_client_for_this_event
            ELSE
                SQL Create new row in table {todos} with (
                    {eventId} = current_event_id_in_db,
                    {content} = todo_from_client.content,
                    {checked} = todo_from_client.checked
                )
            ENDIF
        NEXT todo_from_client

        IF existing_event_in_db IS NOT NONE THEN
            FOR existing_todo_item IN existing_todos_in_db_for_event
                IF existing_todo_item.id IS NOT IN processed_todo_ids_from_client_for_this_event THEN
                    SQL Delete from table {todos} WHERE {id} = existing_todo_item.id
                ENDIF
            NEXT existing_todo_item
```

```
        ENDIF
    NEXT event_from_client

    IF existing_project_in_db IS NOT NONE THEN
        FOR existing_event_item IN existing_events_in_db_for_project
            IF existing_event_item.id IS NOT IN processed_event_ids_from_client_for_this_project THEN
                SQL Delete from table {events} WHERE {id} = existing_event_item.id
            ENDIF
        NEXT existing_event_item
    ENDIF
NEXT project_from_client

FOR existing_project_item IN existing_projects_list_from_db
    IF existing_project_item.id IS NOT IN processed_project_ids_from_client THEN
        SQL Delete from table {projects} WHERE {id} = existing_project_item.id
    ENDIF
NEXT existing_project_item

RETURN SUCCESS
END editDatabase


-> API Route for updating user data.
When routed to "/update-data"...
Check the authorisation header using the token_required function
BEGIN updateData ()
  TRY
    data_from_request = get JSON data from payload
    client_version_tag = data_from_request.versionTag
    projects_to_update_from_client = data_from_request.projects
    user_google_id = Extract the Google ID of the user from the JWT

    current_user_and_projects_data_from_db = getUserDataFromDB(user_google_id)
    user_db_id = current_user_and_projects_data_from_db.user_db_id
    current_db_version_tag = current_user_and_projects_data_from_db.user_version_tag
    existing_projects_from_db = current_user_and_projects_data_from_db.projects

    IF user_db_id = NONE THEN
        RETURN HTTP 400 with JSON {"error": "User identified by JWT does not exist and could not be created."}
```

```
        ENDIF

        IF NOT current_db_version_tag = client_version_tag THEN
            RETURN HTTP 409 with JSON {"error": "Client data is outdated. Please refresh to get the latest data."}
        ENDIF

        edit_result = editDatabase(user_db_id, projects_to_update_from_client, existing_projects_from_db)

        IF edit_result contains "ERROR" THEN
            RETURN HTTP 400 with JSON {"error": edit_result}
        ELSE
            new_app_version_tag = generate a new version tag
            SQL Update table {users} SET {versionTag} = new_app_version_tag WHERE {id} = user_db_id
            RETURN HTTP 200 with JSON {
                "message": "Data updated successfully.",
                "newVersionTag": new_app_version_tag
            }
        ENDIF
    EXCEPT Exception as e
        RETURN HTTP 500 with JSON {"error": "Error processing request."}
    ENDTRY
END updateData ()

-> Begin the server is locally ran.
IF ran as individual file THEN
    Start app in Flask development server
ENDIF
```

https://genta.live > Server pseudocode

```
Import necessesary packages

-> Serve the page

When routed to "/"...
BEGIN home ()
    response = serve the index.html page
    RETURN response
```

```
END home ()


-> Begin the server if locally ran

IF ran as individual file THEN
    Start app in Flask development server
ENDIF
```

https://genta.live > served JS pseudocode

- Only really includes the logic to handle JSON as DOM depends on HTML/CSS.

```
-> fetch the user data.
BEGIN loadUserData ()
    TRY
        response = Fetch data from genta-api.online/get-data
        IF response is successful THEN
            userDataJSON = response
        ELSE
            ERROR ! Failed to fetch user data !
        ENDIF

        store userDataJSON.users.versionTag in local storage

        renderUserData (userDataJSON)
    EXCEPT
        ERROR ! There was an error ! + return captured error
    ENDTRY
END loadUserData ()

-> temporary, client-side only ID.
BEGIN generateTempId ()
    Generates a temporary ID
END generateTempId ()

currentUserData = create dictionary as a memory of the JSON in client
```

```
BEGIN renderUserData (userDataJSON)
    currentUserData = userDataJSON

    -> Ensure all items have a temporary ID if they dont have a server-assigned ID.
    FOR project = first item in currentUserData.projects TO last item STEP one item
        IF project.id = None THEN
            -> Using __ signals that the ID is private (not public).
            projects.__tempId = If project.__tempId does not exist then replace it with generateTempId ()
        ENDIF

        FOR event = first item in projects.events TO last item STEP one item
            event.__tempId = If event.__tempId does not exist then replace it with generateTempId ()
            FOR todo = first item in event.todos TO last item STEP one item
                todo.__tempId = If todo.__tempId does not exist then replace it with generateTempId ()
            NEXT todo
        NEXT event
    NEXT project

    Render all projects, events, and todos based on the currentUserData.
END renderUserData (userDataJSON)


-> User edited an existing project.
BEGIN handleProjectTitleChange (projectIdOrTempId, newTitle)
    project = Find all currentUserData.projects that have projectId OR __tempId = projectIdOrTempId
    project.projectTitle = newTitle
END handleProjectTitleChange (projectId, newTitle)

-> User added a project.
BEGIN handleAddProject (newProjectData)
    -> New projects do not have an id.
    newProjectData.__tempId = generateTempId ()
    Append newProjectData to currentUserData.projects
END handleAddProject (newProjectData)

-> User deleted a project.
BEGIN handleDeleteProject (projectId)
    Remove any project projectId = id or __tempId
```

```
END handleDeleteProject (projectId)

-> User add event to existing project.
BEGIN handleAddEvent (projectIdOrTempId, newEventData)
    project = Find all currentUserData.projects where projectIdOrTempId = id or __tempId
    newEvent = newEventData
    newEventData.__tempId = generateTempId ()
END handleAddEvent (projectIdOrTempId, newEventData)

-> User changed title of ane vent.
BEGIN handleEventTitleChange (projectIdOrTempId, eventIdOrTempId, newTitle)
    event = Find project where currentUserData.projects.id OR currentUserData.projects.__tempId = projectIdOrTempId
    event.title = newTitle
END handleEventTitleChange (projectIdOrTempId, eventIdOrTempId, newTitle)

-> User collapsed or opened an event.
BEGIN handleCollapsedEvent (projectIdOrTempId, eventIdOrTempId, collapsedState)
    event = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event.collapsed = collapsedState
END handleCollapsedEvent (projectIdOrTempId, eventIdOrTempId, collapsedState)

-> User deleted an event.
BEGIN handleDeleteEvent (projectIdOrTempId, eventIdOrTempId)
    project = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event_to_remove = Find event project.events.id OR project.events.__tempId = eventIdOrTempId
    Remove event_to_remove from project.events
END handleDeleteEvent (projectIdOrTempId, eventIdOrTempId)

-> User made new todo.
BEGIN handleAddTodo (projectIdOrTempId, eventIdOrTempId, newTodoContent)
    project = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event = Find event where project.events.id OR project.events.__tempId = eventIdOrTempId

    newTodo = checked: false, content: newTodoContent, __tempId: generateTempId
    Add newTodo to event.todos
END handleAddTodo (projectIdOrTempId, eventIdOrTempId, newTodoContent)

-> if the user checks/unchecks a todo item.
BEGIN handleTodoCheckChange (projectIdOrTempId, eventIdOrTempId, todoIdOrTempId, idChecked)
```

```
    project = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event = Find event where project.events.id OR project.events.__tempId = eventIdOrTempId
    todo = Find todo where event.todos.id OR event.todos.__tempId = todoIdOrTempId
    todo.checked = isChecked
END handleTodoCheckChange (projectIdOrTempId, eventIdOrTempId, idChecked)

-> if the user's todo content changes.
BEGIN handleTodoContentChange (projectIdOrTempId, eventIdOrTempId, todoIdOrTempId, newContent)
    project = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event = Find event where project.events.id OR project.events.__tempId = eventIdOrTempId
    todo = Find todo where event.todos.id OR event.todos.__tempId = todoIdOrTempId
    todo.content = newContent
END handleTodoContentChange (projectIdOrTempId, eventIdOrTempId, todoIdOrTempId, newContent)

BEGIN handleDeleteTodo (projectIdOrTempId, eventIdOrTempId, todoToDelete)
    project = Find project where currentUserData.projects.id OR currentUserData.project.__tempId = projectIdOrTempId
    event = Find event where project.events.id OR project.events.__tempId = eventIdOrTempId
    REMOVE todo in event where event.todos.id OR event.todos.__tempId = todoToDelete
END handleDeleteTodo (projectIdOrTempId, eventIdOrTempId, todoToDelete)

BEGIN sendUpdatedData ()
    currentVersionTag = get 'versionTag' from local storage
    payload = JSON {
        versionTag: currentVersionTag,
        projects: currentUserData.projects
    }

    TRY
        response = Fetch genta-api.online/update-data with {method: POST, body: payload}

        IF response was ok THEN
            Set versionTag in localstorage with responseData.newVersionTag
        ELSE IF response = HTTP 409 THEN
            -> This means that the data on the client is outdated.
            ERROR ! Data has been updated by someone else. Please reload. !
        ELSE
            ERROR ! Failed to update data !
        ENDIF
```

```
    EXCEPT
        ERROR ! Error sending data to server for update ! + caught error
    ENDTRY
END sendUpdatedData ()
```

Desk checking

NOTE: Due to the size and complexity of my project, my desk checks have been significantly reduced to just be generalised possibilities of my backend. They are designed not for display but just so that I'm able to run through and validate if my program is functional and if I need to make any changes to it.

If the user doesn't exist on the db getUserDataFromDB (user_google_id)

User_goog-e_Id	User_record	User_db_id	User_version_tag	New_version_tag	projects_from_db	Assembled_projects_data	RETURN
newID	NONE	NONE	NONE				
		1	123-123	910-123	NONE	[]	{ "user_id_id":1, "user_version_tag":123-123, "projects":[] }

If the user does exist with 1 project inside getUserDataFromDB (user_google_id)

User_goog-e_Id	User_record	User_db_id	User_version_tag	New_version_tag	projects_from_db	Assembled_projects_data	RETURN
newID2	{id: 2, versionTag: 'v1'}	NONE	NONE				
		2	v1	v2	[{id: 10, ...}]	[]	
						[...]	{ "user_id_id":2, "user_version_tag":v1, "projects":[...] }

A user with 1 event to add and 2 todos to update editDatabase (…)

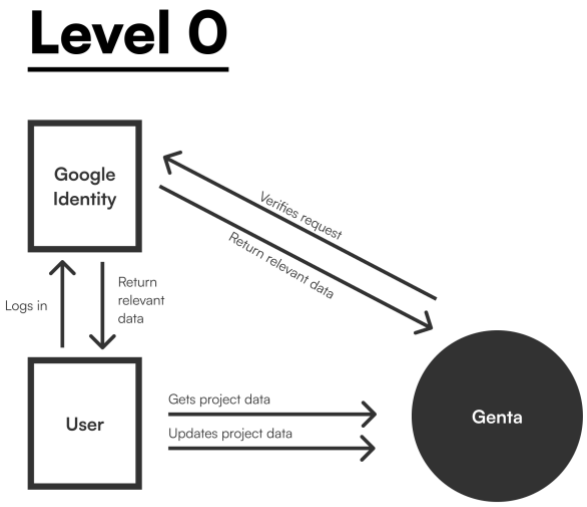
Processed Ids	Project_in_client	Existing_project_in_db	Existing_project_in_db	New_project_id		SQL

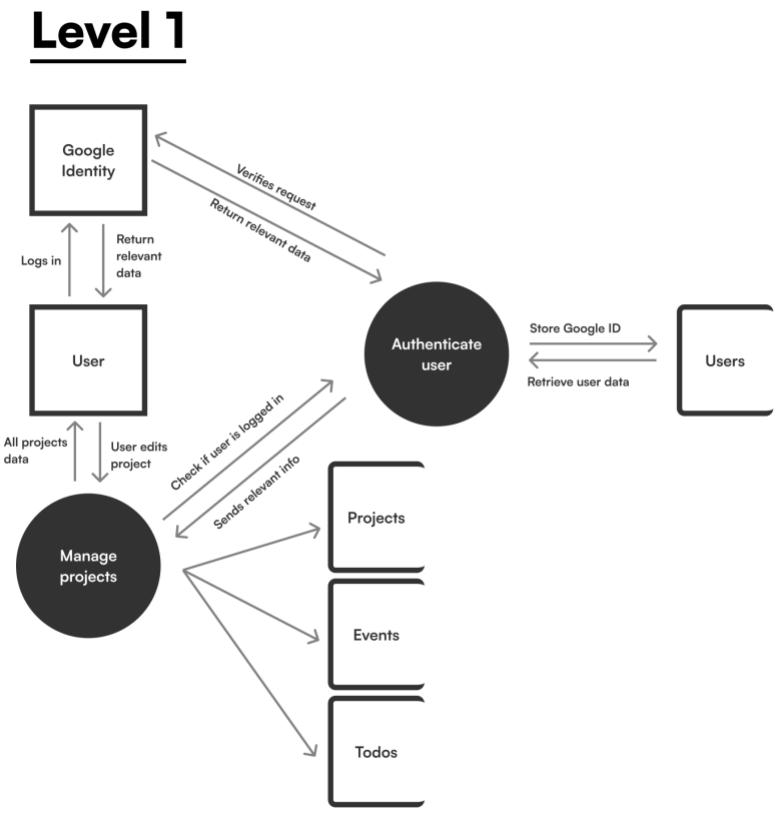
{}		NONE	{[...]}	1		1, projTitle, 2025-01-01

Events_from_client	Processed ids	Existing_event_in_db	Current_event_id_in_db		SQL
[...] 1 to add	{}	None	None		
		[...]	1		{ "projectId":2, "title":v1, "collapsed":[...], "dueDate":"2025-01-01", "notes":"", "todoShown":true, "notesShown"false: }

Todo_for_client	Existing_todos_in_db_for_event	Processed ids	Existing_todo_in_db	current_todo_id_in_db	todo_from_client.id	SQL
[...] 2 to update	[...] 1 exists	{}	None		1	
			[...] 1 match	1		{ "eventId":1, "content":"demo", "checked":false, }
			None		4	ERROR Todo with ID 4 to update not found under 1

Data flow diagram





Data dictionary

genta-api.online / **getUserDataFromDb()**

Variable	Type	Description	Example
user_google_id parameter	String	The google id of the user for quering the db.	“1092831203...”
users_with_id	List	A list (or typically a single item) of all the records matching user_google_id	[{“id”: 1,...},...]
user_foreign_key	Integer	The primary key (id) of the user from {users} table	1
projects_with_id	List	All projects from {projects} table which have user_foreign_key	[{“id”:12, “projectTitle”:”im sick”, ...}...]
project_data	List	A list of dictionaries representing projects with their children events and grandchildren todos	[{..., “projectTitle”:”im sick”, ..., events: [...]}, ...]
i, j, k	Integers	Loop counter variables	1,2,3...
Project_dict	Dictionary	A dictionary representing a single projects dat	[{..., “projectTitle”:”im sick”, ..., events: [...]}
events_with_id	List	All event records from the {events} table with project_data[i].id	[{... "title": "influenza or wtv", "collapsed": false, ...}, ...]
event_dict	Dictionary	A dictionary representing a single event’s data	[{... "title": "influenza or wtv", "collapsed": false, ...}
todos_with_id	List	All todo records from the {todos} table	[{... “checked”:false, “content”:”get tablets from pharmacy”},...]
todo_dict	Dictionary	A dictionary that represents a single todo	{... “checked”:false, “content”:”get tablets from pharmacy”}

genta-api.online / **getData ()**

Variable	Type	Description	Example
user_google_id	String	The google ID extracted from the JWT of the authenticated user	“122938982...”
project_data ↗ **getUserData FromDb ()**	List	A list of dictionaries from the getUserDataFromDb function	[{..., “projectTitle”:”im sick”, ..., events: [...]}, ...]
response_data	Dictionary	A data structure prepared for the JSON response	{“user_id”:1, “versionTag”:..., “projects”: [...]}
jsonified_response	String (as JSON)	The response_data as a JSON string	‘{“user_id”:1, “versionTag”:..., “projects”: [...]}’

genta-api.online / **editDatabase ()**

Variable	Type	Description	Example
user_foreign_key parameter ↗ **updateData**	Integer	The primary key (id) of user	1
data_to_update parameter ↗ **updateData**	List	List of project dictionaries with potential changes and new objects that are sent from the client	[{..."projectTitle": "New Title"}, {"projectTitle": "Brand New Project", "dueDate": "..."}]
existing_user_data parameter ↗ **updateData**	Dictionary	The current user data fetched from the database, used for comparison	{“user_id”:1, “versionTag”:..., “projects”: [...]}
project_to_update	Dictionary	A single project dictionary	{... “project title”:”new proj”...}
existing_project	Dictionary (or None)	Project from existing_user_data.projects that matches project_to_update.id	{... “project title”:”new proj”...} Or None
project	Dictionary	Each (one at a time) project in existing_user_data.projects	{... “project title”:”new proj”...}

event_to_update	Dictionary	A single event	{... “eventTitle”:”new event”...}
existing_event	Dictionary (Or None)	Event from project_to_update.events that matches event_to_update.id	{... “eventTitle”:”old event”...} Or None
event	Dictionary	Each (one at a time) project in project_to_update.events	{... “eventTitle”:”new event”...}
todo_to_update	Dictionary	A single todo from event_to_update.todos	{"id": 2, "checked": false, "content": “my new task"}
todo	Dictionary	Each (one at a time) todo in event_to_update.todos	{"id": 2, "checked": false, "content": “my task"}
result	String	Indicates success or error	ERROR or none

genta-api.online / **updateData ()**

Variable	Type	Description	Example
data_from_request	Dictionary	The JSON payload received from client request	{"versionTag": "389023a", "projects": [...]}
client_version_tag	String	The version tag sent by the client	389023a
data_to_update	List	The projects part of data_from_request containing the data to be updated in the databse	[{"id": 101, "projectTitle": "New Title"}, ...]
user_google_id	String	The google ID of the user extracted from the JWT	09128013821...
users_with_id	List	All user records from the database matching user_google_id. Expected to be just one	{"id": 1, "user_google_id": "109...", "versionTag": "302..."}
user_foreign_key	Integer	The primary key (id) of the user	1
newVersionTag	String	A newly generated version tag to be assigned after a successful update	9203b

existing_user_data	Dictionary	Current user data fetched from the database via getUserDataFromDb	{"user_id": 1,...
result	String	Indicates success or error	ERROR or none

genta.live / **all**

I am going to group similar variables together as this code is very repetitive. This will make the data dictionary more concise and allow me to adhere to the page limit imposed on this assessment task.

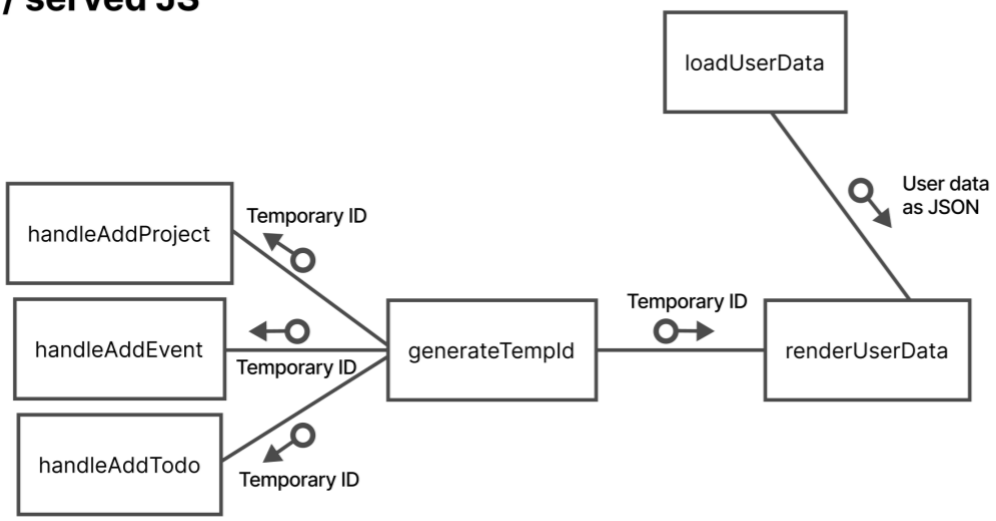
Variable	Type	Description	Example
currentUserData & userDataJSON	Object	The main object for holding all user related data typically as a JSON, which is modified via user interatctions	{"user_id": 1, "versionTag": 190239", "projects": [{"id": 101, "projectTitle": "SEN proj", "__tempId": 092348, "events": [...]}, ...]}
projectIdOrTempId, eventIdOrTempId, todoIdOrTempId, projectID, and todoToDelete	String or Integer	An identifier to locate a specific project, event, or to do. Can be a client side __tempId or server side id.	Server: 209 Client: 20394a
__tempId	String	A client side generated ID for objects that are created within a session and hence do not have any server identifier.	20938402938a

Class diagrams (not applicable to my project)

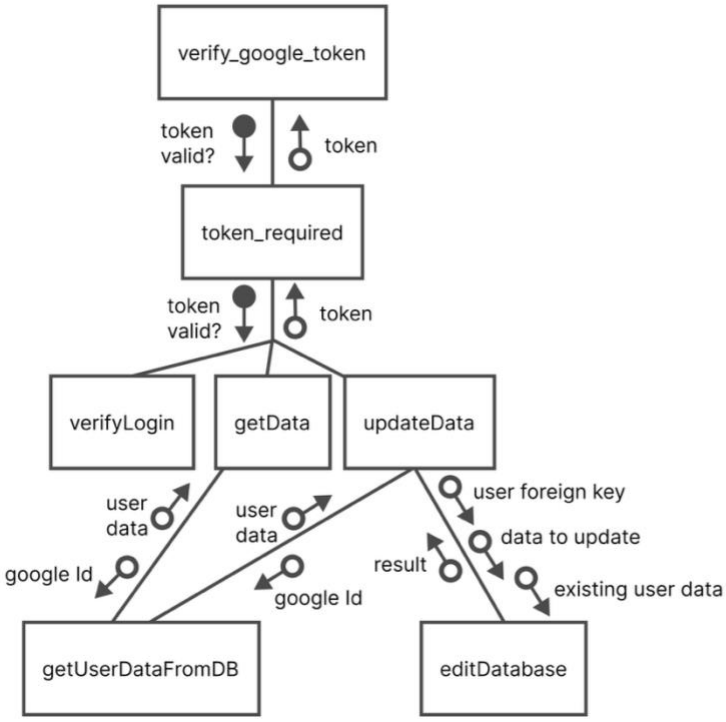
There are no classes defined or used in my design. My codebase employs individual functions to encapsulate specific logic designed to manage data and does not use classes.

Structure charts

genta.live
/ served JS

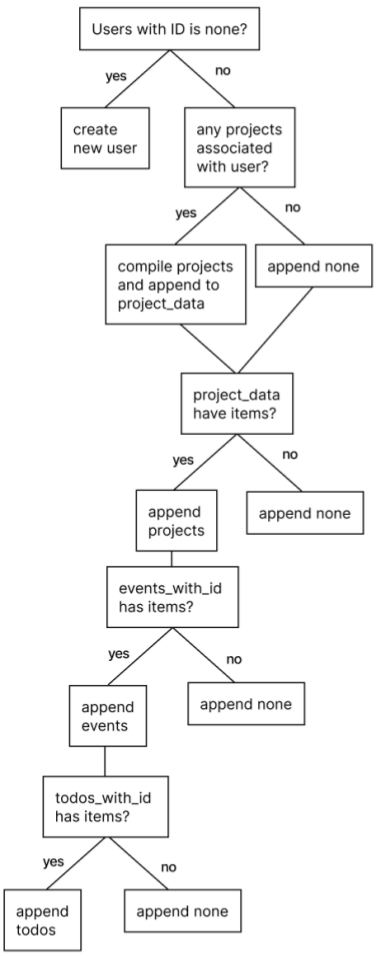


genta-api.online
/ server

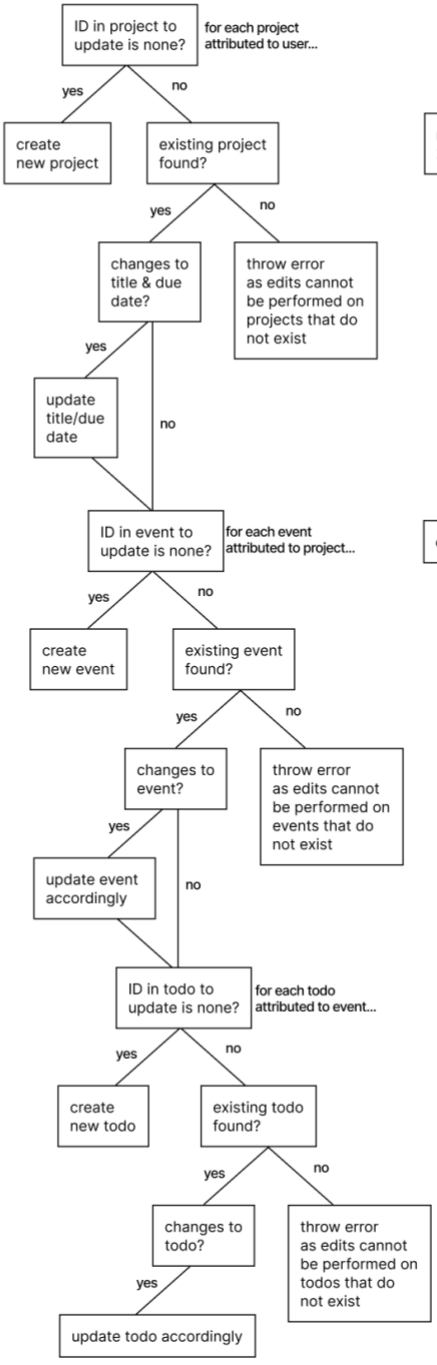


Decision trees for **Genta API**

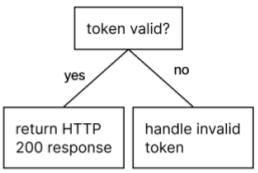
get user data from db function



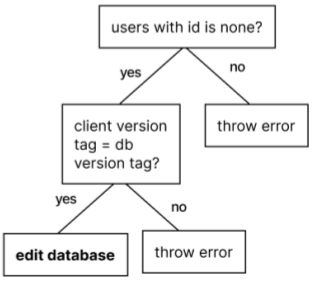
edit database function



token validation for all api routes

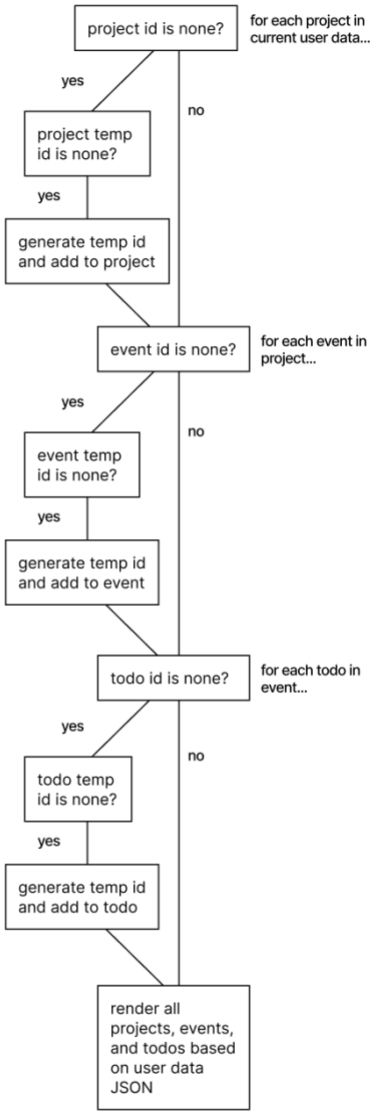


update database API route



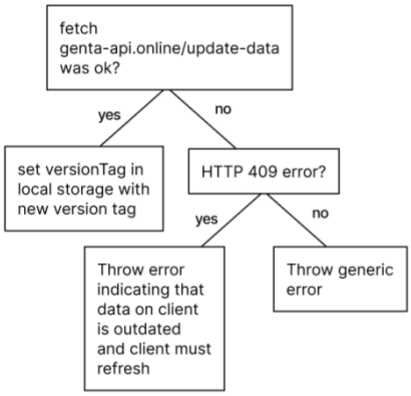
Decision trees for **Genta Client**

render user data function



load user data function

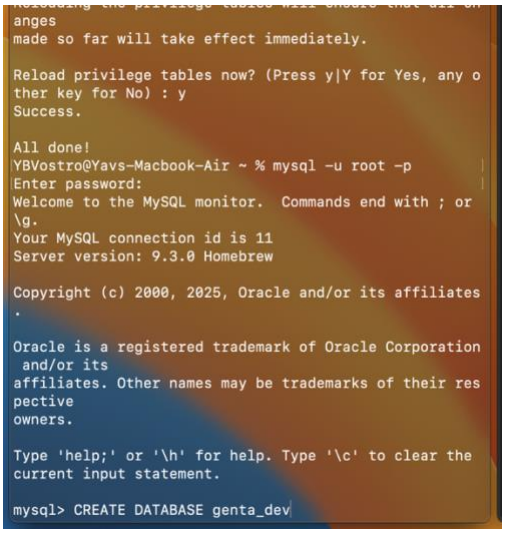
send updated data function


fetch genta-api.online/update-data was ok?
yes
no
set versionTag in local storage with new version tag
HTTP 409 error?
yes
no
Throw error indicating that data on client is outdated and client must refresh
Throw generic error

Solution to software problem

Now that I have my systems modelling complete, I need to actually program my webapp. The first setup would be to set up a proper local developer environment that I can effectively use to develop on my local machine, then push that to production.

Step one would be to set up a mysql server on my local machine. To do this, I used a guide[31].


anges
made so far will take effect immediately.
Reload privilege tables now? (Press y|Y for Yes, any o
ther key for No) : y
Success.
All done!
YBVostro@Yavs-Macbook-Air ~ % mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or
\g.
Your MySQL connection id is 11
Server version: 9.3.0 Homebrew
Copyright (c) 2000, 2025, Oracle and/or its affiliates
.
Oracle is a registered trademark of Oracle Corporation
and/or its
affiliates. Other names may be trademarks of their res
pective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the
current input statement.
mysql> CREATE DATABASE genta_dev

1. Install mysql with homebrew
    a. brew install mysql
2. Start the mysql server
    a. brew services start mysql
3. secure the mysql installation (follow on screen prompts)
    a. mysql_secure_installation
4. Create database user for development environment
    a. mysql -u root -p
5. Create the database in mysql
    a. {mysql} CREATE DATABASE genta_dev;
6. Add new yser and grant privileges (replace mylocaluser and mylocalpassword with the desired username/password)
    a. {mysql} CREATE USER ‘mylocaluser’@’localhost’ IDENTIFIED BY ‘mylocalpassword’;
    b. GRANT ALL PRIVILEGES ON genta_dev.* TO ‘mylocaluser’@’locahost’;
    c. FLUSH PRIVILEGES
        i. Flush privileges is deprecated but its good to do it anyway.
7. Exit mysql
    a. EXIT;
8. Update my .env file on my development database
    a. DB_HOST=localhost
    DM_PORT=3306
    DB_USER=mylocaluser

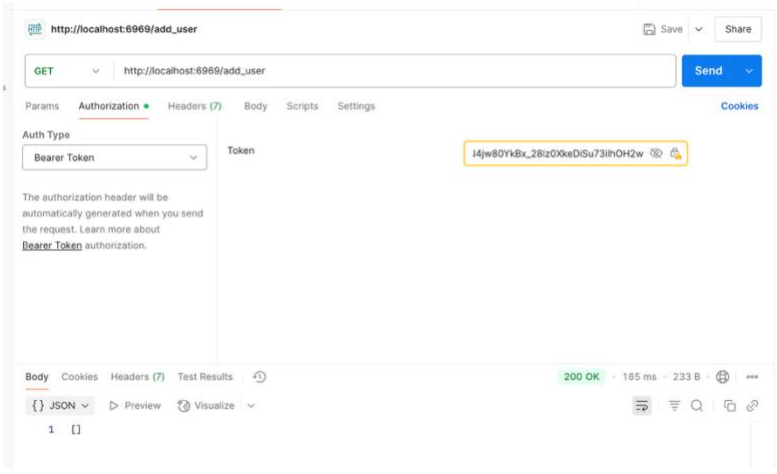
[31] CheapVPS (2024) Installing MySQL on macOS with Homebrew, Accessed May 25, 2025
https://atlassc.net/2024/03/11/installing-mysql-on-macos-with-homebrew

DB_PASSWORD=mylocalpassword
DB_NAME=genta_dev

9. If youre using a .sh file for your environment variables, you might need to update it on your system:
    a. source environmentvar.sh
10. When finished with mysql for the meantime, I can stop mysql using:
    a. brew services stop mysql
11. and restart using
    a. brew services start mysql
    b. log into mysql → *mysql -u root -p;*
        i. or log into a specific user → *mysql -u mylocaluser -p*
    c. Show all databases
        i. SHOW DATABASES;
    d. Select a DB to work with (you don't need to exit mysql to USE a different db, just run this command again)
        i. USE database_name;
    e. Exit the database
        i. EXIT;

CREDENTIALS REDACTED

Then, I used Postman to ping my local API. Because my local API requires sign in with google, I:



- Went to https://genta.live
- Signed in with Google
- Opened inspect element > console
- Copied the "Encoded JWT ID Token"
- And put it inside postman to ping http://localhost:6969 authorisation header → auth type: bearer token → Token: <Encoded JWT ID token>

This returned a HTTP 200 response, but the JSON didn't have anything inside (shown in screenshot above as a "[]". This is because I haven't set up the database yet.

Jump to Table of Contents

I wrote this code to create the database tables, using the source and knowledge learnt from W3Schools that I used in the creation of my database dictionary. From there, I was able to also set the constraints for each item in the table.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    googleId VARCHAR(21) NOT NULL,
    versionTag VARCHAR(255) NOT NULL
);

CREATE TABLE projects (
    id INT AUTO_INCREMENT PRIMARY KEY,
    userId INT NOT NULL,
    projectTitle VARCHAR(255) NOT NULL,
    dueDate DATE NOT NULL,
    FOREIGN KEY (userId) REFERENCES users(id)
);

CREATE TABLE events (
    id INT AUTO_INCREMENT PRIMARY KEY,
    projectId INT NOT NULL,
    title VARCHAR(255) NOT NULL,
    collapsed BOOLEAN NOT NULL DEFAULT FALSE,
    dueDate DATE NOT NULL,
    notes TEXT,
    todoShown BOOLEAN NOT NULL DEFAULT TRUE,
    notesShown BOOLEAN NOT NULL DEFAULT TRUE,
    FOREIGN KEY (projectId) REFERENCES projects(id)
);

CREATE TABLE todos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    eventId INT NOT NULL,
    checked BOOLEAN NOT NULL DEFAULT FALSE,
    content TEXT,
    FOREIGN KEY (eventId) REFERENCES events(id)
);
```

Using the DESCRIBE table; query, I was able to check if each table was correctly formed.

Projects table

```
mysql> DESCRIBE projects;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int          | NO   | PRI | NULL    | auto_increment |
| userId       | int          | NO   | MUL | NULL    |                |
| projectTitle | varchar(255) | NO   |     | NULL    |                |
| dueDate      | date         | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
4 rows in set (0.002 sec)
```

Users table

```
mysql> DESCRIBE users;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int          | NO   | PRI | NULL    | auto_increment |
| googleId   | varchar(21)  | NO   |     | NULL    |                |
| versionTag | varchar(255) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.013 sec)
```

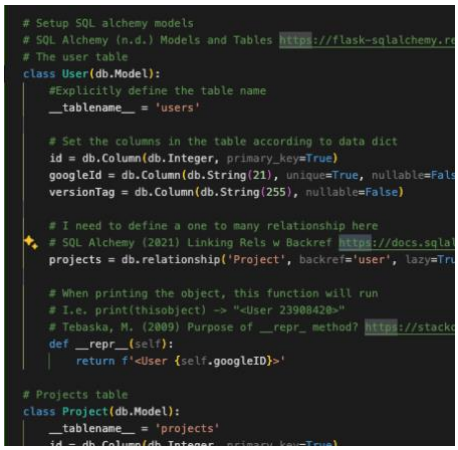
Events table

```
mysql> DESCRIBE events;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int          | NO   | PRI | NULL    | auto_increment |
| projectId  | int          | NO   | MUL | NULL    |                |
| title      | varchar(255) | NO   |     | NULL    |                |
| collapsed  | tinyint(1)   | NO   |     | 0       |                |
| dueDate    | date         | NO   |     | NULL    |                |
| notes      | text         | YES  |     | NULL    |                |
| todoShown  | tinyint(1)   | NO   |     | 1       |                |
| notesShown | tinyint(1)   | NO   |     | 1       |                |
+------------+--------------+------+-----+---------+----------------+
8 rows in set (0.003 sec)
```

Todos table

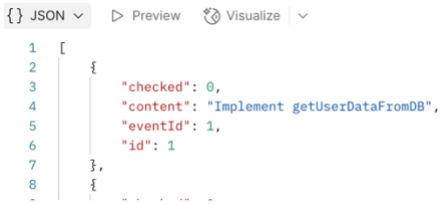
```
mysql> DESCRIBE todos
    -> ;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int         | NO   | PRI | NULL    | auto_increment |
| eventId | int         | NO   | MUL | NULL    |                |
| checked | tinyint(1)  | NO   |     | 0       |                |
| content | text        | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
4 rows in set (0.003 sec)
```

Now, I have to follow my pseudocode to develop my backend.

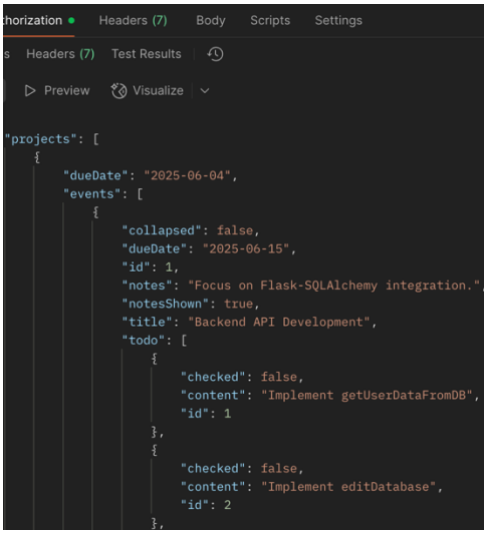

# Setup SQL alchemy models
# SQL Alchemy (n.d.) Models and Tables https://flask-sqlalchemy.r
# The user table
class User(db.Model):
#Explicitly define the table name
__tablename__ = 'users'
# Set the columns in the table according to data dict
id = db.Column(db.Integer, primary_key=True)
googleId = db.Column(db.String(21), unique=True, nullable=Fals
versionTag = db.Column(db.String(255), nullable=False)
# I need to define a one to many relationship here
# SQL Alchemy (2021) Linking Rels w Backref https://docs.sqla
projects = db.relationship('Project', backref='user', lazy=Tru
# When printing the object, this function will run
# I.e. print(thisobject) -> "<User 23908420>"
# Tebaska, M. (2009) Purpose of __repr_ method? https://stacko
def __repr__(self):
return f'<User {self.googleID}>'
# Projects table
class Project(db.Model):
__tablename__ = 'projects'

First, I needed to redefine my old SQL alchemy models to the new current database setup. I learnt from multiple guides including the SQL alchemy docs and stackoverflow posts to assist me in setup. These are shown in the code

Using Postman, I was able to test this!

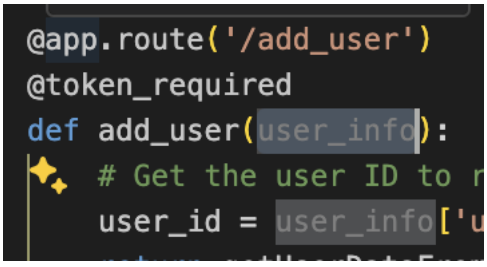


After that, I was able to code the getUserDataFromDB() function. To do this, I input some demo data into my database, which I could retrieve as a JSON in an API call in postman. However This was using a demo user ID




```
return getUserDataFromDB("123456789")
```

To fix this, I needed to get the user's Google ID (aka "sub") for my database. To do this, I edited the decorator function (in the vendor provided code) to pass the user_info into the 'add_user' function/endpoint *requires changing*. This way, when I



update the database with my personal google account's sub (in image):

IMAGE REDACTED FOR PRIVACY

Now the JWT that I pass into the authorisation header will change the data that I get back from the API endpoint because the server retrieves the associated account's sub and uses that to find that specific user's data on the mysql database.

Data returned from API endpoint /add_user when using JWT returned from auth with **y..@gmail.com**	Data returned from API endpoint /add_user when using JWT returned from auth with **y...@gmail.com**

A user is found, so the getUserDataFromDB() function returns the data associated with the account.
 	
"projects": [
{
"dueDate": "2025-06-04",
"events": [
{
"collapsed": false,
"dueDate": "2025-06-15",
"id": 1,
"notes": "Focus on Flask-SQl
"notesShown": true,
"title": "Backend API Develc
"todo": [
{
"checked": false,
"content": "Implemer
"id": 1
},
{
"checked": false,
"content": "Implemer
"id": 2
},
{
	{
	"projects": [],
	"user_db_id": 3,
	"user_version_tag": "7888b2
	}

Few problems with implementing the editDatabase function:

- I needed to compare the date object SQL with the JSON. To do that I needed to convert the date obj to str, and I did that using a guide from geeksforgeeks using the date.fromisoformat() func.
- I wanted a way to check if an instance is a type of instance without having to constantly set a var so I can get it closer to my pseudocode. A friend showed me the isinstance() method which I learnt more about in this w3schools article.
- Database committing for keys required something called session.flush()... or not, I do not know anymore. It's deprecated I think. If it works it works.
- User version tag wasn't properly incorporated in the client side.

NOTE: I NEED TO EDIT GENTA.LIVE TO REFLECT GENTA-API.ONLINE ENDPOINT CHANGES

After spending too long coding this was the result of the get-data function via an api endpoint!

Here's an example of what the get-data function outputs! (partially reduced to ...)


```
{
  "projects": [
    {
      "dueDate": "2025-06-04",
      "events": [
        {
          "collapsed": false,
          "dueDate": "2025-06-15",
          "id": 1,
          "notes": "Focus on Flask-SQLAlchemy integration.",
          "notesShown": true,
          "title": "Backend API Development",
          "todo": [
            {
              "checked": false,
              "content": "Implement getUserDataFromDB",
              "id": 1
            },
  ...
```

```
                ],
                "todoShown": true
            },
  ...
        "id": 2,
        "projectTitle": "Project2"
    }
  ],
  "user_db_id": 1,
  "user_version_tag": "init-ver-tag"
}
```


Now the editDatabase function can take that and examine all the changes within the JSON, editing the database accordingly.

To test this, I used white-box testing methods.

WHITE-BOX TEST *editDatabase()*		
What I need to test	*How I did it*	*Results*
Handling a new user with /get-data	Signed into genta.live with Emily's account and got the JWT to input into the user	Success! Functions as normal
Adding a new project Adding a new event Adding a new todo	By using a new user's account (my sister's) I pasted in an existing user's JSON with 'null' in each ID field, hence appearing like the user wants to add this to their database.	This works! Everything functions (almost) normally and adds to the database. However, I have found a problem: within a session, when the user adds a new thing, the id in the JSON is set to "null". This indicates that there is something without an ID and hence the server needs to create a new item for that user. Within a session, the client sticks with its own idea of what the JSON *should* look like, and the robust-ish versionTag methodology tries it's best to ensure that it's correct. The problem is that the session has no (planned) way to update its JSON.

		Ironically, this is technically not a showstopper because when the API recognises that something is missing, it thinks that the user has deleted it and hence will remove that from the database. When an ID is still null and the expecting ID is not present, the database will just delete the missing one and add the new one. All this means is that the server will experience more load as it needs to regenerate the user's items. Also, the IDs on the server side will spike as each item is allocated a new ID. Maybe we can make it so that the client fetches the data while running or that recommendations to refresh are given after a set amount of time.
Deleting a project Deleting an event Deleting a todo	I used my sister's account and sequentially deleted a todo, then event, then project.	Deleting everything works! This includes cascading deletions too; as in when you delete a project it's associated events and todos are deleted, along with when you delete an event, it's associated todos are deleted.
Editing project items: - Title - Due date Editing event items: - title - collapse state - due date - notes - show todos - show notes Editing todos: - checked state - content	I went in and simply just edited a bunch of data in the JSON and looked at the mysql db.	I noted that turns out user_db_id is sent to the client and can be edited, but alas I only use the id given by the database and not the client. Otherwise, everything works as normal

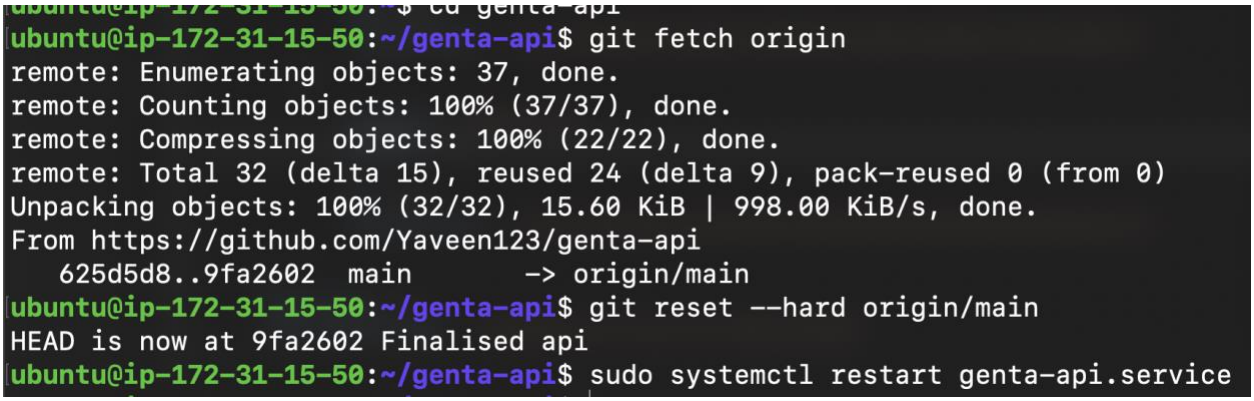
New users accessing /edit-database endpoint	Signed into genta.live with my sister's account and got the JWT to input into the user.	*Technically* it works as the user has no associated versionTag, therefore the server will always say "Client data is outdated. Please refresh to get the latest data." But this does mean that I'm not actually checking if the user exists before editing their db. I added that check in, and now it says with User does not exist on the database. Run /get-data endpoint to create a user. a 404
Prevent JSON with db ids that don't belong to that user from editing that row in SQL	I simply added a JSON from some else's account into a new user's account as a request in update-data	Success! It'll fail to find the project with the associated ID. There may be a possible security vulnerability where children nodes can be used to find out their parent node but I'm not entirely sure how that would work.

I am now confident that I can deploy my API.

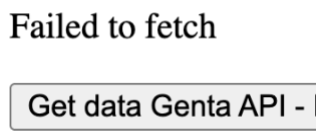
I started by repeating the same steps of creating my database on my local machine to creating the database on my API server.

IMAGE REDACTED FOR PRIVACY

I fetched the updates from git onto my ec2 instance


ubuntu@ip-172-31-15-50:~/genta-api$ git fetch origin
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 32 (delta 15), reused 24 (delta 9), pack-reused 0 (from 0)
Unpacking objects: 100% (32/32), 15.60 KiB | 998.00 KiB/s, done.
From https://github.com/Yaveen123/genta-api
625d5d8..9fa2602 main -> origin/main
ubuntu@ip-172-31-15-50:~/genta-api$ git reset --hard origin/main
HEAD is now at 9fa2602 Finalised api
ubuntu@ip-172-31-15-50:~/genta-api$ sudo systemctl restart genta-api.service

And now genta.live is (correctly) broken!



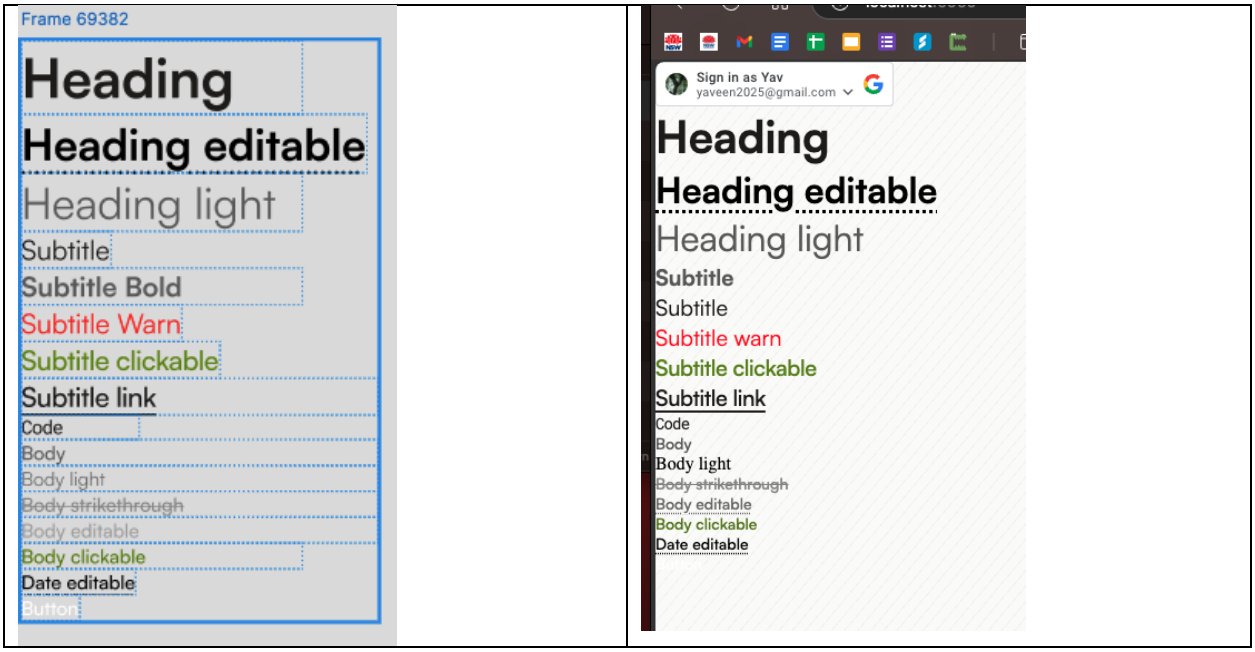
This means that I have to redesign genta.live. I simply added a few new buttons which ping the API correctly, and as you can see it works!

Genta Alpha
Welcome to Genta Frontend
Welcome to this quick demonstration of the updates to https://genta-api.online/
Genta operates on 3 main API routes that you're able to test.
You're signed in! Try fetching the data from the API now.
{"projects":[],"user_db_id":1,"user_version_tag":"18c7ca4b-5027-408c-bc25-b664125596
Test /verify-login
Test /get-data
Test /update-data *doesn't work
Sign out of Google

Now I can work on my frontend properly. I started by designing a demo sign in screen that users have to get through.

The first thing I did was clean out the whole existing index.html file and then replace it with a group of fonts that I needed to match using the CSS. I compiled all the fonts in Figma, and then in dev mode I copied the CSS for each font and put it into typography.css. I also learnt proper name handling for files, using _... for files that are to be combined into main.css, and inside that using the .block__element—modifier system.

Figma	Web



Then I was able to code in my solution for the login area.

I learnt that in order to have padding be able to not add into the existing width, I have to use something called box-sizing that forces an inclusion of the padding (and border) in an element's total width and height[32]. This makes using Figma dev mode generated CSS SO much easier because things work expectedly with box-sizing.

IMAGE REDACTED FOR PRIVACY

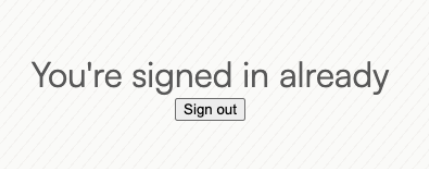
[32] W3schools (n.d.) CSS Box Sizing, Accessed 28 May 2025 https://www.w3schools.com/csS/css3_box-sizing.asp

For now, Genta doesn't check if the JWT is valid using the https://genta-api.online/verify-user route. Instead, it just checks if the JWT *exists* in the local storage, a major flaw in my existing design.

After adding in that functionality (by hooking up the existing old API GET request to the login function), I was then able to deploy it to Github.

- I did find a problem where index.js/window.onload() function attempts to perform a JWT crosscheck with genta-api.online but MAYBE verifies it correctly. I literally have no idea and I'm not *really* bothered trying to fix it because at any other stage in the

I started with the titlebar, I spent 2 days (!!) trying to be able to add things without everything breaking. While basic opening and closing was functional, as soon as I tried to animate it, there was a critical bug where when the header was open, items would push the header (as it is positioned absolute) out of the viewport, hiding the hamburger icon. If the icon stayed visible, the JS toggle would still misfire, closing when opening and opening when closing.



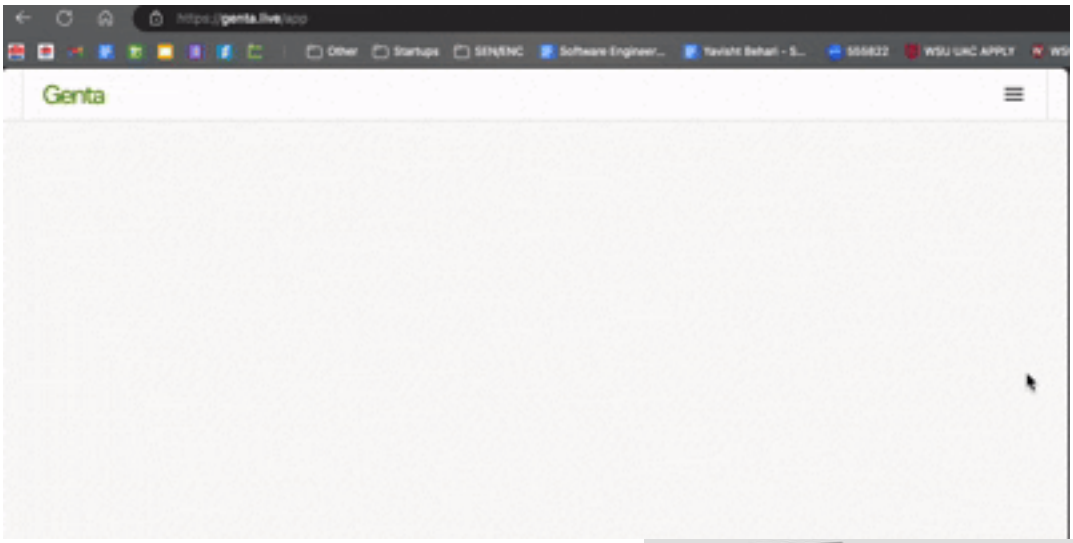
I fixed these bugs by reworking the JS and spending hours playing with the CSS, eventually making a function:

```
let heightToSet = 50 + document.getElementsByClassName("header__link").length * 68;
```

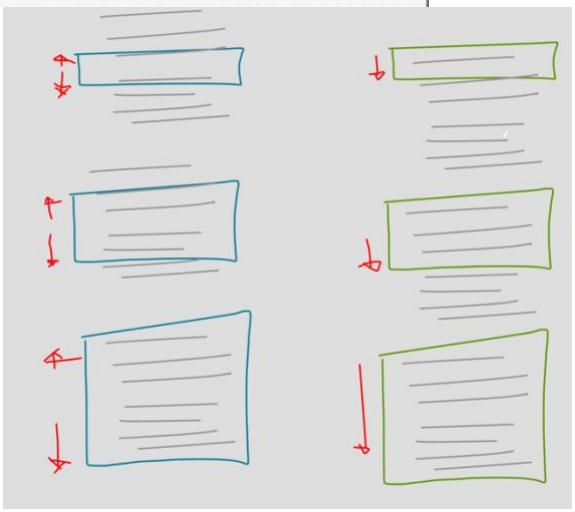
Which mathematically calculated the height the containing element – "header" had to expand to in order to fit the links inside. Then, it'll smoothly transiton it's maximum height from 50px to the required to fit. JS then came in and hid (opacity:0) the inner components and switched the icon. After that, I added a hover-animations and semantic open/close functionality using sources outlined in the code. I learnt about CSS variables[33], and how to make marching ants[34] (sources & functionality outlined in code [static/css/components/_titlebar.css], description shortened due to page limit).

[33] runofthemillgeek (2017) Set parent's background colour, accessed May 30 https://stackoverflow.com/questions/45580218/css-set-color-to-parents-background-color

[34] MDN (n.d.) repeating-linear-gradient() accessed may 30 https://developer.mozilla.org/en-US/docs/Web/CSS/gradient/repeating-linear-gradient

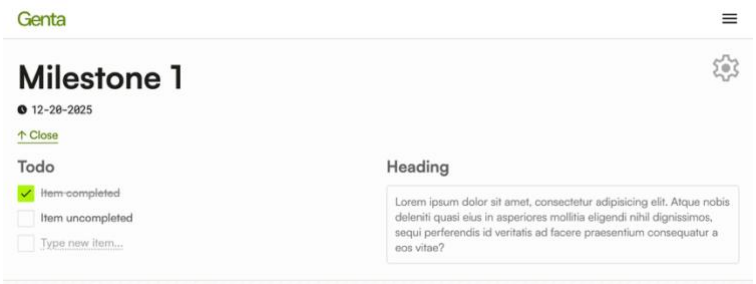


Later I was able to fix the bug of the header going out of the screen when closing (which was caused by the content justify being changed from flex-start to center, making it so that when the header expanded, the content in the middle would go to the center instantly and then position itself outwards when it scaled.

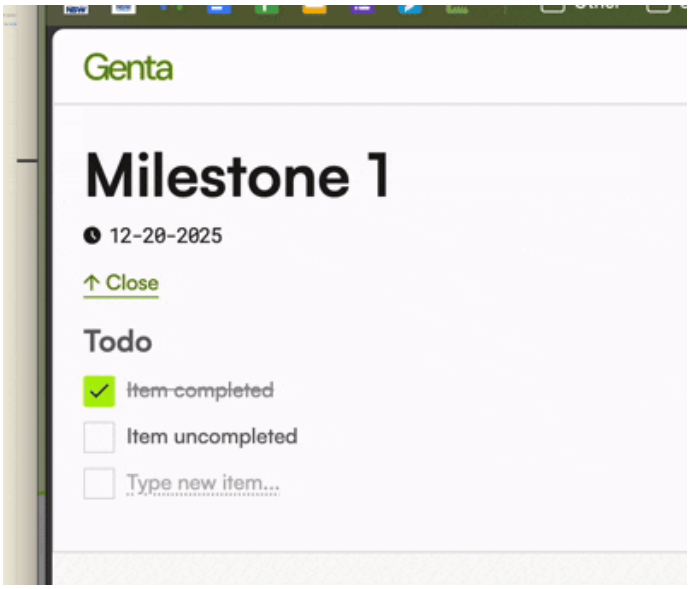


This is illustrated in the diagram on the side, blue = buggy, green = fix.

From that I coded a base, non-functional version of the events just to get all the classes working. It took time but overall it wasn't buggy at all.

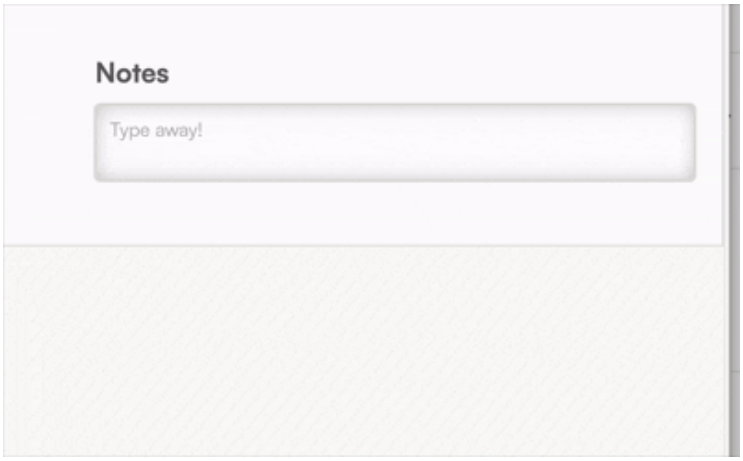

Genta
Milestone 1
12-20-2025
Close
Todo
Item completed
Item uncompleted
Type new item...
Heading
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Atque nobis deleniti quasi eius in asperiores mollitia eligendi nihil dignissimos, sequi perferendis id veritatis ad facere praesentium consequatur a eos vitae?

I also procrastinated and added a tiny confetti effect on the non-functional complete icon, using the canvas-confetti library[35].



I learnt how to do this from the documentation - I added an on-click event listener to all unselected to do icon buttons, and then launched the confetti effect according to the documentation with at that specific point with settings that I tweaked.

I also randomised it, so there's a chance of getting rainbow, and a change of getting Genta green!



Before going back to do the to-dos, I quickly go the notes area up and running. It was surprisingly complex, so here's what I learnt:

- Large textboxes can't just be a common input field, they're actually something called "textarea" elements[36].
- Placeholder text can actually be styled using the "::placeholder" pseudo-element[37].
- Textareas don't respond to height scaling based on fit-content. Instead, I learnt that we must use JavaScript whenever the page loads or the user updates, we have to force the page to look at the text content and determine the height that the div has to be[38].

---

[35] Catdad (2024) Canvas-confetti documentation, Accessed may 31, 2025 https://github.com/catdad/canvas-confetti

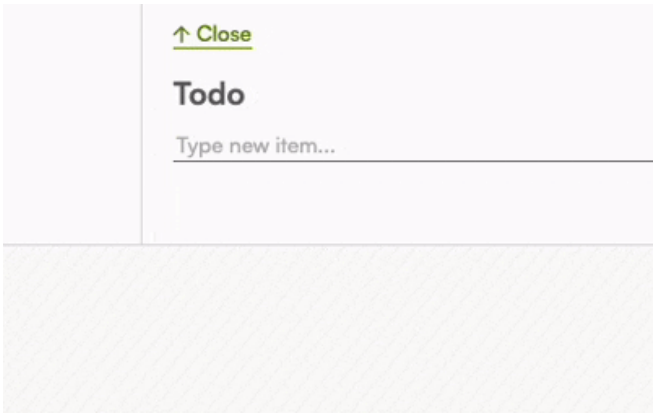
[36] w3schools (n.d.) HTML <textarea> tag, Accessed May 31, 2025 https://www.w3schools.com/tags/tag_textarea.asp

[37] MDN (n.d.) ::placeholder, Accessed May 31, 2025 https://developer.mozilla.org/en-US/docs/Web/CSS/::placeholder

[38] "DreamTek" (2014) creating a text area with auto-resize, Accessed May 31, 2025 https://stackoverflow.com/questions/454202/creating-a-textarea-with-auto-resize

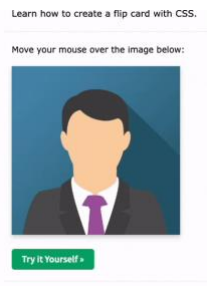
- I also learned that we are able to disable spellcheck so whenever the user has clicked off the text box, the spellcheck is disabled and when spellcheck is reenabled [39].
    - In order to do this, I learnt that there is something called a blur event[40] that goes hand-in-hand with a focus event.
- I learnt that it's possible to edit or completely remove the focus border[41].

I made the todos functional next. To do this, I needed to learn a few things:



- Reading and preventing spaces or empty enter items is really easy[42].
- In order to create the todo based on the enter, I learnt what a "keydown" event is[43].
- My todo items were vulnerable to XSS, and to prevent that I learnt that you can just create an element and then add the text in using the textContent in standalone and that should be not vulnerable to XSS[44].
- Hover events in JS can be coupled as mouseenter and mouseleave[45].

I also added functionality to the delete button and the expand/close item.



~~Next, I need to code the edit-event functionality. I'm going to follow this guide from W3Schools (shown in the gif) to make a flip-card effect, where on the back theres going to be a HTML form to edit info~~... of which I have scrapped because it causes too many issues.

[39] Wendelin, E (2008) Disable spellchecking on HTML textfields, Accessed May 31, 2025 https://stackoverflow.com/questions/254712/disable-spell-checking-on-html-textfields

[40] MDN (n.d) Element: blur event https://developer.mozilla.org/en-US/docs/Web/API/Element/blur_event

[41] CEich' (2010) How to remove focus border (outline) around text/input boxes? Accessed May 31, 2025 https://stackoverflow.com/questions/3397113/how-to-remove-focus-border-outline-around-text-input-boxes-chrome

[42] "bobbyg603" (2019) How to detect string which contains only spaces? Accessed 1 June 2025 https://stackoverflow.com/questions/10261986/how-to-detect-string-which-contains-only-spaces

[43] MDN docs (n.d.) Element: keydown event Accessed 1 June 2025 https://developer.mozilla.org/en-US/docs/Web/API/Element/keydown_event

[44] Gallegos, J (2023) XSS: Using 'innerText' or 'textContent vs InnerHTML' Accessed 1 June 2025 https://medium.com/@xjgalleg/xss-using-innertext-or-textcontent-vs-innerhtml-e60ac660167e

[45] MDN (n.d.) Element: mouseover event Accessed 1 June 2025 https://developer.mozilla.org/en-US/docs/Web/API/Element/mouseenter_event

Instead, I used a simple transition animation to show that the edit box was open/closed. I also just duplicated an event to create the main project item, date object, and add event item.

There were a few bugs that really bugged me out:

- Webkit on iOS/iPadOS wanted to be all different and didn't work well with my transitions, and everything looked massive. I compromised for just setting a large overflow maximum.
- For the longest time I just developed one event and completely forgot that IDs aren't re-usuable (probably because I was too busy trying to understand guides on websites), so I had to change a significant portion of my JS to use classnames and iterate over then (i.e. document.querySelector('classname').forEach((element) => {...} instead of just a simple document.getElementById)
- It took SO long to get the transition between edit states to work, every time the edit-box would disappear or just go away and leave it's children behind (background colour not applying on descendants)
- Because I was reusing the event element for Today and +Click to add event, I had to use dummy classes to keep my event listeners happy.

There were many more but none of them were truly noteworthy.

Now I have to get syncing to the cloud working. This is the largest part. I've broken it up into test cases:

Test	Info
Passing gID into dataMgr	Functions correctly. There were a few issues like having to wait for gId to fire correctly, but other than that it was pretty easy. I constructed a little function to test the api to do that. data-manager.js ▶ {message: 'Successfully signed in with Google.'}
Id generation	Functions correctly. I used the crypto.randomUUID from this post. and verify google ID TEST CASE 2 PASSED -> Able to generate version ID, example: 4926dbe7-1462-45cf-931d-5e77b7ed3834 data-manager.js:115

Get user data and pass as JSON	This was relatively straightforward until it came to assigning temporary IDs. First I realised that I didn't even have an ID system so I used this one. Then I realised that I had to actually assign the IDs. After a while, I was able to craft a function that can be recursively called to parse projects -> events -> todos if they're not assigned an ID.

Rendering data	This is a large one, and I kept getting very overwhelmed about it, so I asked an online friend for a little help. My existing approach was to render data and then rely on closest-to-descendent placement to "guess" the location of an element within the JSON and hope that nothing goes wrong. Otherwise, I would generate a client ID for every element, create a dictionary of those elements, and then scrub the entire project for that specific ID.
	
	However, he said that these approaches were (*awful and*) unnecessary and instead I can just bake in data to the HTML elements using data-* attributes.
	
	You know how HTML elements have the tags like
	 Hello world
	He told me to make my own tags like
	 Hello world
	
	This makes referencing in my JS very easy. For example, to create a todo, I'd js have a single function:
	createTodoItemDOM(projectId, eventId, todoData) {
	itemDiv = document.createElement('div');
	itemDiv.className = 'ev__todo-item'
	itemDiv.dataset.eventId = eventId
	itemDiv.dataset.projectId = projectId
	...
	}
	
	Where .dataset refers to creating the data-* tag via DOM.
	
	He told me that I should create a new file and put my DOM creation logic into there, and that I had to essentially shift my logic JS to the file.

JC also taught me some neat variable assignment called short-circuit evaluation! It makes if statements more compact.

I kept running into issues, such as forgetting that arrays can be empty, and these were frustrating because JS would just give up every time.

Other errors I ran into

```
loadInitialUserData > Error in loading data: TypeError: document.elementElement is not a function
data-manager.js:203
```

I also kept making stupid typos. This one took me 20 MINUTES (!!!) to fix because JS just doesn't want to tell me what the error was.

```
const eventHeader = document.elementElement('div');
eventHeader.className = 'event__header';
```

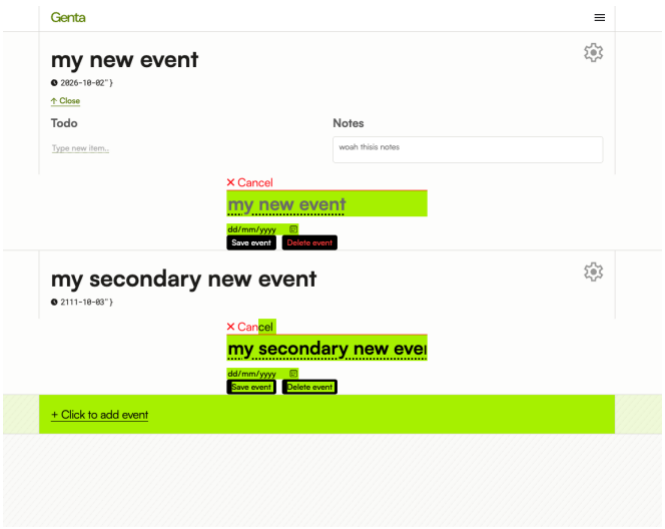
(elementElement not createElement)

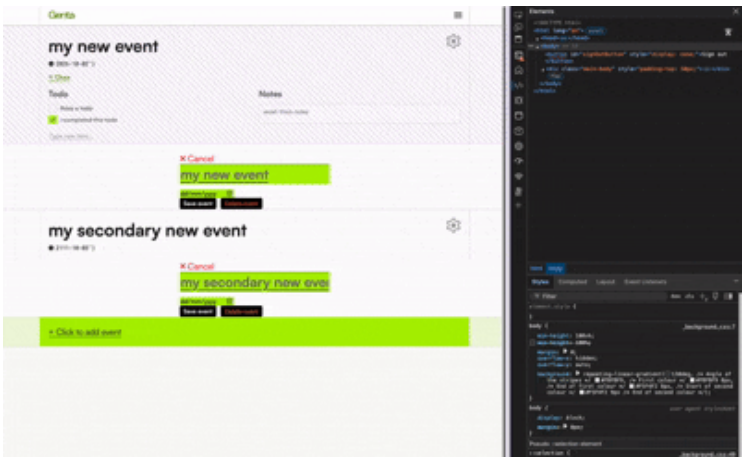
```
loadInitialUserData > Error in loading data: ReferenceError: Cannot access 'expandCollapseButton' before initialization
data-manager.js:203
```

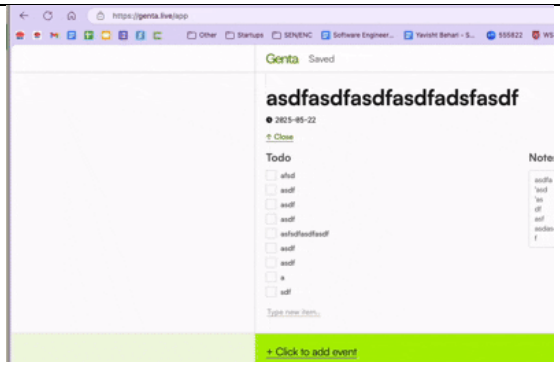
I kept getting confused between what to put for the variables and for expandCollapseButton I split it into *text and *type but missed that completely for a specific section

```
loadInitialUserData > Error in loading data: ReferenceError: addEventPlaceholder is not defined
data-manager.js:203
```

I kept testing with plceholders but kept forgetting to port stuff over like here I moved everything to a proper addEvent var but still used the placeholder var



	After wraping event listeners I kept finding problems with my placement or things were hiding when they shouldn't be, for example the whole content thing was hiding when it should only be the section.  After these were fixed, I was able to successfully load from genta-api and display items. To make my life easier I just got gemini to generate some sample data and I put that into my db.
Autosaving data	*Note: Due to page limits and stress this section of the report is retroactive* I started by adding project switching. I had many issues with variable assignment and wrapping my head around actively rendering active data based on local-storage. Then I had to generate the JSON. This took numerous different tries; and even though the implementation sucks but my project generated the JSON by looking into the DOM and hoping that everything matched up. The implication of this would be something like below:

	 Where changing the DOM directly saves into the autosave. *Yes, my project reallllllllly sucks*. After getting the update-data function working, I needed to just deploy and touch up! I added event editing, event saving, and project adding. After this, During the testing process, I was able to fix numerous bugs and push Genta to the final stages! These would include (but are not limited to): - Saving was erratic for items like todos - Autosave error handling was hidden from the user, which I fixed using the autosave prompt at the top - Fixing delete project functionality - Fixing issues with project switching - Removing unnecessary band-aid reloads. - Malformed project IDs when updating caused 400s In the server.

After this, I cleaned up the code (removed useless console logs and added better descriptions for otherwise sparsely explained sections) and got it ready for submission!

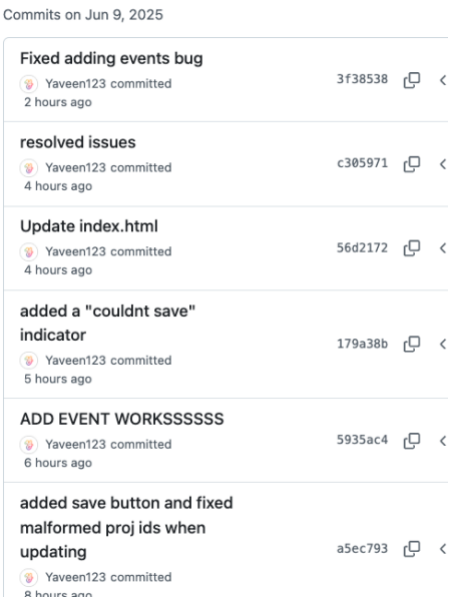
Jump to Table of Contents

Version control

Implement version control when developing a software engineering solution.

Version control was handled entirely by Github. Each commit was assigned a different ID and outlined the changes to the project, encapsulating how the project changes overtime.

Using Github hit two birds with one stone – it allowed me to easily deploy my projects to their EC2 instances, and also handled version control and snapshot management in one go.



Testing and evaluating

Code optimisation

Students explain the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

- **Dead code elimination**
  - Dead code elimination is when “dead code” is removed from a program. This “dead code” refers to any code that never actually reaches execution, so it just takes up space and execution time. By removing dead code, the size of the software is reduced, and the hardware doesn’t have to execute as many lines.
  - In Genta, at many points in my frontend design, I commented out code that no longer is required or perceptually didn’t made a difference to the final result.
- **Code movement**
  - Code movement[46] is when statements are strategically moved outside of a loop when the result of that computation doesn’t change during that loop. It improves speed because if a computation is always providing the same

[46] GeelsForGeeks (2023) What is code motion? Accessed Jone 10, 2025, https://www.geeksforgeeks.org/what-is-code-motion/

result over and over, it's more computationally efficient to just calculate it once (outside the loop) and use that value.
    - While in the backend this was simple, the frontend was difficult because I'm still relatively new to inline functions and callback functions, along with odd scoping requirements and the general elasticity of JS. Definitely, looking back, there are many points in my frontend where I could've significantly optimised how the loops function, however due to the inconsistent (and downright messy) nature of Genta's frontend design and the time limits imposed on this project and other subjects, I wasn't able to optimise this well.
- **Strength reduction**
    - Strength reduction refers to replacing computationally expensive operations with weaker, less expensive ones (reducing the strength of the operation to improve speed). For example, multiplication might be replaced with a combination of adding and subtracting[47]. This is typically used in scenarious like induction variables, where a variable is used to control the iteration of a loop (typically i).
    - Genta rarely uses induction variables in its backend, and mathematical operations in the frontend are kept to a minimum. Due to this, I can't really mathematically improve my codebase.
- Common sub-expression elimination (CSE)
    - CSE is where an expression or subexpression that has been computed before avoids recalculation by instead favouring a stored single calculation and reusing that.
    - This technique requires a very controlled and consistent understanding of a codebase and the innerworkings of each function of a codebase, along with an extremely granular plan of how the codebase functions. This is not realistic for a student.
- Compile time evaluation – constant folding and constant propagation.
    - Constant folding is when expressions that consist of only a constant value at compile time can be computed at compile time with that single constant value[48].

[47] GeeksForGeeks (2025) Induction variable and strength reduction, https://www.geeksforgeeks.org/induction-variable-and-strength-reduction/

[48] Opengenus (n.d.) The need for constant folding and constant propagation in compiler design, https://iq.opengenus.org/constant-folding-and-propagation/

- Constant propagation on the other hand is the process of replacing references to a variable with it's known constant value. I.e. if we know that x = 10, and we compute y = x + 5, constant propagation tells us to use y=10+5, replacing the reference variable with the constant.
- For Genta, this is overkill and constant assignment in general was a constant headache. These are concepts that are definitely worth keeping in mind in the future, but weren't applicable to Genta.

Testing Methods

Apply methodologies to test and evaluate code

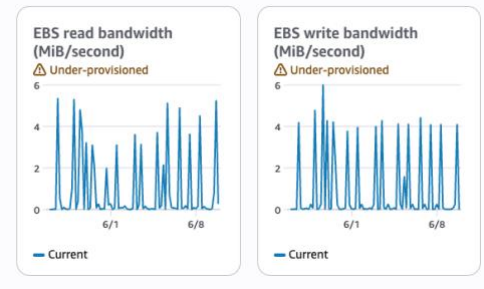
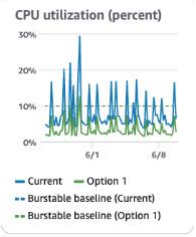
Method	**Applicability**	**Reasoning**
Functional testing	Applicable	Functional testing is when you check if each function of the system correctly does what it's supposed to do, regardless of speed[49]. To do so, I'd need to have a set of defined expectations of what that function will do and then test to see if the function actually did it. I was able to do a lot of functional testing throughout the entirety of my project, especially during the times where I performed whitebox testing on my API. The whitebox testing enabled me to identify errors on easily by having a predefined goal of what the subroutine or processes was supposed to do, and then evaluate the output of that.
Acceptance testing	Not applicable	Acceptance testing is a formal testing process which determines whether a software meets the needs of the client or end user. This is typically performed not by me but my client[50]. Unfortunately, my client cannot perform a formal acceptance test within the time constraints of this project.
Live data	Applicable	This is when you use actual, real world production environments to test.

[49] GeeksForGeeks (2025) Functional Testing – Software Testing, accessed 9 June 202, https://www.geeksforgeeks.org/software-testing-functional-testing/

[50] GeeksForGeeks (2024) Acceptance testing. – software testing, accessed 9 June 2025, https://www.geeksforgeeks.org/acceptance-testing-software-testing/

		Live data was used as soon as possible. The first instance of live data that was used was through the implementation of the JWT synchronisation on the backend, which allowed me to use real subs from actual Google accounts. After Genta Alpha was able to be deployed, slowly live data took precedence over simulated data, with the goal of being able to use Genta with no simulated data in the end.
Simulated data	Applicable	This is when you use simulated data to test. Simulated data was necessary especially during the earlier days of my front-end development. Because rendering data came before editing projects/events/Todos, having simulated data readily available was the only way to properly figure out of my rendering system was fully functional.
Beta testing	Applicable	This is when a nearly complete version of the software is released to a selected group of actual users (pilot) to give feedback of bugs, usability testing, performance, etc. As a developer, I am not an accurate representation of the end-user, and hence I might not see all the flaws in a software system. Instead of unreliably trying to put myself into someone else's shoes, I was able to launch a beta testing program for Genta and have a few beta testers on board so I can polish Genta before release. These fixes are visible in the testing section.
Volume testing	Not applicable	Volume testing is when you test a software application with a large amount of data to determine the system's capacity and how it performs under high data loads[51]. This is extremely technical, and requires knowledge that I do not have. If I were to volume test, the outcome would generally just be that "under high loads Genta slows down". It is plausible to say that Genta API is not the fastest or most optimised system, however the effort of optimising it would be infeasible compared to my other subjects, The closest I can get to genuine volume testing is understanding that my AWS compute instance for

[51] GeeksForGeeks (2024) Volume testing, accessed 9 June 2025, https://www.geeksforgeeks.org/volume-testing/

		Genta API sometimes takes a while to push out the large JSON that it needs to the client.  Otherwise, on normal load Genta uses around 17% of it's CPU each request, which is fine and not a cause for concern. This means that my backend is optimised well enough for basic loads. 

Test data tables

Testing data used/generated based on path and boundary testing

Boundary test

Because Genta lacks proper validation limits (the limit is imposed by HTTP request size or when the database fills up to its max 20gb allocation, both of which aren't coded into Genta itself), the only boundary test that I could possibly think of is the date field.

Variable	Max	Min	Default	Expected	Actual	Reason for inclusion
Date	31-12-9999	01-01-0001	1-1-2025	Test 1: 01-01-0000 Test 2: 31-12-9999 (cant enter anything higher) Test 3: 12-12-2031	01-01-0001 31-12-9999 21-21-2031	This is the only applicable variable for a boundary test

Path test

Variable/s	Default	Expected Actual	Reason for inclusion
Event data placeholders	N/a	If project card: titlePlaceholder = "Untitled project" saveButtonText = "Save project" deleteButtonText = "Delete project" saveButtonClass = "save-project" if event card: titlePlaceholder = "Untitled event" saveButtonText = "Save event" deleteButtonText = "Delete event" saveButtonClass = "save-event-button"	Event data placeholders always switch based on whether the card is a project card or a regular event card. These are crucial for identifying what goes where and where to put it.
Event data collapsed placeholders	Type: down Text: Expand	If project card: collapsed Type: down, text: expand If project card not collapsed: Type: up, text: close.	Collapsed/expanded states need to be controlled by something
Todo existence path	[] (empty)	Fills up with todos based on demand. i.e. [{content: “test”, checked: false}]	A card can have anywhere between 0 to 1 or more todo items.
Callback func for delete	N/A	If project: Window.deleteProjectLocalStorage() If event: markDataHasChanged() + outerBox.remove()	The callback function for a delete object changes based on whether the associated card is a projet or event.
Finding events	N/A	Normal case -IN: (validProj, ‘...’) OUT: return event Project card case - IN: (validProj, ‘...’) OUT: return proj Null card - IN: (null, ‘...’) OUT: null	Cases for searching for a valid event change based on the information available to Genta.

		Can't find event - IN: (validProject, null) OUT: null	

Methodology to test and evaluate code

Unit, subsystem, and system testing	These three types of testing create a hierarchy which is ordered by granularity[52]. **Unit testing** is when the software is broken down into "units" which represent the smallest possible part to test in that system. This ensures that the unit works correctly in isolation. For example, in Genta, verifying a user's sign in has only output and was tested in isolation to see if it's functional. The next level of testing is **subsystem testing**[53] where a large software system is broken down into smaller modules/subsystems and then is typically tested through the flow of the data through that. However, this isn't a well-established concept. A possible example is how Genta runs 3 test cases to ensure that data collected from the server has been correctly captured and rendered without errors. The final level is **system testing**, where a testing process evaluates the final software to ensure that it meets its requirements. The system is tested to see if it works as expected. An example of this is the end-user testing feedback section of this report, where users gave in their feedback and possible problems that they experienced. It's important to note that these don't work alone, instead a system takes each three and works backwards/forwards to ensure that the system works correctly.
Black, white, and grey box testing	**Blackbox testing** is when the tester has no knowledge of the internal code structure, and they treat the software as a "black box" (cannot see inside); and therefore, are able to highlight errors due to noticing deviations in the

[52] GeeksForGeeks (2025) Difference between unit testing and system testing, accessed June 10 2025, https://www.geeksforgeeks.org/difference-between-unit-testing-and-system-testing/

[53] "GhostCat" (2017) What is subsystem testing for software testing, accessed June 10, 2025, https://stackoverflow.com/questions/42471751/what-is-subsytem-testing-for-software-testing

	normal functioning of the system. Like if the "saving..." message on the top left of Genta displays an error correctly if there's a problem with saving. **Whitebox testing** on the other hand is when the tester has full knowledge of the internal code structure. I did a very explicit whitebox test on my API before working on my frontend in order to ensure that my API worked expectedly! **Grey box testing** is when the tester has limited knowledge of how a system works. It combines the two, for example someone might know the database schema, but they'll have to test it through Genta API; they don't know the code within the API. JC did many grey box tests to help me fix problems in my backend.
Quality assurance	QA attempts to prevent problems by ensuring that the software development process itself stays aligned to it's true goals in delivering a product that is of high, shippable quality. QA attempts to make the system robust enough to avoid bugs and other issues that can affect the usability and safety of the system. The purpose of QA is to build quality into the product from the beginning, referring to the whole quality process. Genta's QA was one of the first things my client and I did together.

Analysis of solution against quality success criteria

Comparing actual output with expected output

Quality criteria	Met?	Analysis
USABILITY The user experience for Genta needs to be intuitive and easy to work with in order for efficient navigation. Users need to be able to set their projects, create new events/tasks/milestones, and to be able to edit them accordingly with a	Yes	Many users expressed positive sentiment about Genta's UI/UX. At no point did any user struggle with the UI, and most found it intuitive. While one person got confused about the difference between events and projects due to how they appear the same, every other tester was able to understand the purpose of Genta. From a UX/UI standpoint, users were able to create new events/tasks/projects and were able to edit them with a minimum click-depth of 2; a respectable amount.

minimal number of clicks/taps.		
FUNCTIONALITY Corresponding to Genta's MVP, Genta needs to be able to provide a platform that reliably creates, edits, and saves projects, events, and the data within them.	Yes, to a nuanced Extent	Other than filtering tasks, every aspect of Genta's MVP were met. Some were renamed: - Tasks, deadlines, and meetings, were merged into "events" - "Context" was dropped for a simple notes section - "Workspaces" were rebranded to projects While aspects like not being able to distinguish between events that represent tasks, deadlines, or meetings, or having no need for filtering, or no descriptions for projects, David and I feel that these requirements didn't hold up anyway when put to the test of actual UI/UX design, and Genta still embodies the core of what he wanted.
PERFORMANCE Genta should respond quickly and smoothly to user interactions even when internet connection speeds are slow. This includes the smooth navigation of tasks and projects, on both desktop and mobile.	Yes	Other than two complaints of the slowness of project adding performance of Genta was significantly improved over the duration of the development time. Significant optimisations in the front-end codebase made rendering much faster, and this allows for a snappy and responsive experience. Many users enjoyed the small animations (and requested more) that added to the feel of the system. Ignoring bugs, this was consistent across mobile and desktop. When internet is slow, Genta still performs as saving happens in the background – users do not notice if their connection is stuggling as the saving process is hidden from the user.
DATA INTEGRITY User data, including projects and events, should be stored securely and retrieved accurately and reliability. Data needs to be sent from the database to the client securely, through methods like HTTPS encryption in transit.	Yes	Genta uses Sign in with Google to extract a sub from a secure JWT (authorisation token). This process is incredibly secure and industry standard. Genta-API is designed so that the only data that a user can tamper with is their own data, as it only sends/allows updates from IDs linked to that user's Google account. Data is encrypted during transit using standard HTTPS. HTTPS is a requirement for Sign in with Google.

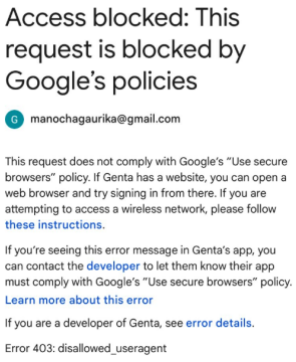
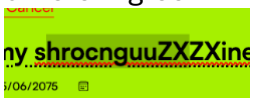
SECURITY Sign in with Google needs to be implemented according to Google's official, latest guidelines, ensuring that the IdP is implemented securely and reliably.	Yes	Although problematic at times, Sign in with Google was implemented with almost entirely up to date guidelines and procedures. The "almost" comes from Genta not implementing Google's new FedCM variant of Sign in with Google, however the current model is definitely sufficient for what it needs to do. Sign in with Google has been set up so that the only information that Genta receives from the user's account is it's publicly accessibly sub, significantly reducing the amount of data that could possibly be lost during a data breach.
BOUNDARIES Genta needs to adhere to the originally set boundaries, like no AI, and no collaboration. This ensures that Genta remains focused to my client's original goal.	Yes, almost	- Genta has no AI features - Genta has no collaboration features - Genta is designed to be open to how the user wants to work, and doesn't force conformity into certain productivity frameworks - Genta has small note fields - Genta's links feature was dropped due to complexity and time constraints - Planning tasks on Genta doesn't really work, you just put your events in and go straight ahead! - Genta only uses Sign in with Google - Genta works on both mobile and desktop without compromise to its features (excluding one android bug)

Analysis of feedback

Gather feedback and analyse it

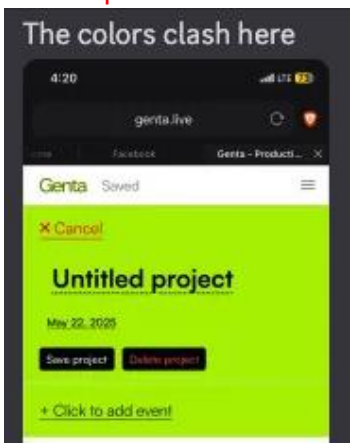
Developing a report to synthesise feedback

User	**Platform**	**Feedback/issues**	**Response**
NAME REDACTE	MacOS, Chrome	**Feedback statement:**	*I wanted colour coding sooo bad*

D FOR PRIVACY	1 issue	*"Hmm thwre isnt rly much feedback I can give I think it's rly good* *Maybe u can allow the user to colour code the different projects? Idk if that's hard for u to do just a suggestion idk* *Or like u can add collaborators to that? Or thwres a function that allows u to add links to take u to another website to Google docs or wtv relating to that?* *Idk I'm just teynna say smth so u have smth to work from 😭BUT ITS SO GOKD GOOD JOB 🥳🥳"* **Issues:** Cannot resolve. When opening Genta on Instagram mobile, Instagram uses it's own (possibly WebView) browser that causes this error:  Access blocked: This request is blocked by Google's policies manochagaurika@gmail.com This request does not comply with Google's "Use secure browsers" policy. If Genta has a website, you can open a web browser and try signing in from there. If you are attempting to access a wireless network, please follow these instructions. If you're seeing this error message in Genta's app, you can contact the developer to let them know their app must comply with Google's "Use secure browsers" policy. Learn more about this error If you are a developer of Genta, see error details. Error 403: disallowed_useragent	*but man its DIFFICULT to code this project* *You sent and collaborators is just extremely, extremely difficult* *You sent links was literally in the plan i just removed it cuz i was dying of stress 😭 BUT YEAH THANK YOUUU* I'm not sure what the solution to this really is other than to use a different browser, simply because if Google is blocking the request then it knows something is wrong.
NAME REDACTED FOR PRIVACY	Windows 11, Microsoft Edge 2 issues	**Feedback statement:** *Nice!* **Issue 1:** Resolved. Selecting project text results in a green background, but that is the same green that I use for the edit-box, so it appears the same.	*Thanks!* Resolved! I changed the selection text background to a different green.  ny shrocnguuZXZXine /06/2075

		Issue 2: Cannot resolve. No temp-id was generated for the new untitled project, causing a "Couldn't save" error.	I didn't have enough time or data to debug this. There could be a problem with the JSON being malformed but I'm not sure. If someone else runs into a similar error I could be able to debug it.
NAME REDACTED FOR PRIVACY Attempt 1	Samsung Internet on Android in Secret Mode 1 issue	**Issue:** Cannot resolve. There was another problem with Samsung internet. When loading, Google sign in would get stuck at https://accounts.google.com/gsi/transform	I'm not too sure what the problem is. I couldn't find a solution online, so my best bet is Samsung internet's Secret Mode enhanced tracking prevention, blocking Google from making redirects. This isn't something I can fix.
NAME REDACTED FOR PRIVACY attempt 2	Windows 11, Chrome	**General feedback:** *The genta.live website seems to be well put together, I quite like the UX/UI elements, and as a beta build I do believe that it can significantly evolve, into an even more practical productivity tool then it is already. The simplicity of the project allows for it to not only serve as a*	*Thank you so much!!* *This feedback is awesome, and I'd definitely consider adding more customisation/dark mode in the*

		lightweight alternative to other more "bloated" apps of a similar category, but it also seems to be reliable and efficient, as it can quickly autosave new edits to the Notes, Events, and Project tabs of the site. Whilst the project is of a great standard for a beta build, I do believe that it could benefit from the use of increased cybersecurity, a dark mode functionality, and maybe even a font change/customisation option. *I think Yav's project is of a high standard thus far and can become a practical option in the productivity category.*	*future, and cybersecurity is definitely a focus :D*
NAME REDACTED FOR PRIVACY	Brave on iOS 1 issue	**Feedback 1:** Cannot entertain in a timely manner; low priority. *You should add an add project button to the main page when you don't have any projects rather than just saying "create a new project by clicking the __ button"* **Feedback 2:** Cannot entertain; requires codebase redesign *Creating a brand new project should be instantaneous, no reason for the user to have to wait for loading.* **Feedback 3:** Cannot entertain; requires codebase redesign *Maybe add a create project ui where you can immediately specify all the basics like the project title*	*Yes totally agree, but due to project overrun times and constraints, I can't add that rn.* *Ah yes, that's definitely what's supposed to happen, but I had issues with API conditions and optimistic loading. In the future, I'm definitely going to see if I can change that.* *That's what I was also going for, but due to complexities in my implementation it's quite difficult. Namely because new projects don't open after creation, so opening the "Untitled project" and instantly getting hit with edit mode would be off-putting.*

		Feedback 4: Cannot entertain, client specification  **Feedback 5:** Cannot entertain, cost blow-out. *Saving takes a while — I would add an animation like a loading sign to make it feel faster. Saving is really slow…* **Issue 1:** Resolved. *The project title doesn't immediately update when you change it in the projects menu, you have to reload the view/refresh for it to update* **Feedback 6:** Cannot entertain; UI fundamental problems. *When you press click to add event, you should make the input field*	*Hmm, that's by design, my client wanted it that way. But I'll raise that concern with my client asap!* *Autosaving like that is by design, and takes 5 seconds after an edit. This is because my server is under provisioned and I don't have the money to up-spec it lol.* *This is quite the bug! I'll see if I can resolve that.*

		automatically selected so you can start typing without an extra click **Feedback 7:** Cannot entertain in a timely manner; low priority. *An animation to close/expand would be cool*	*i'll take that on board, but personally I don't like that (if more people want it I'll add it.* *I agree.* This is isn't a priority tbh.
NAME REDACTED FOR PRIVACY	Windows 11, Chrome	**Issue:** Resolved Adding multiple events creates duplicate events.	This is due to adding multiple event listeners and I was able to resolve.
NAME REDACTED FOR PRIVACY	Windows 11, Opera GX	**Feedback:** Cannot entertain in a timely manner; low priority. Enter to save and escape to cancel Feedback: Autosave is awesome and works well	This is cool but would take too long to implement,

Analysis:

Generally, there was significant noteworthy positive sentiment. During testing, many users liked the simplicity of the UX/UI and by extent of that enjoyed Genta's perceived reliability and efficiency. People definitely loved Genta's autosave feature. For the bugs that were resolved, testers were happy with the results of them.

In terms of negative feedback, some testers encounter issues that couldn't be resolved because of technical complexity, like the intermittent but elusive "Couldn't save" error (with no server response). Possibly due to poor setup, sign in with Google appeared to be very sensitive to the smallest security problems. Other testers also wanted features or improvements that I couldn't add in because it would take too much time, too much money, or goes against the codebase's design.

In conclusion, Genta is generally well received and praised for my design of it's core functionality and UX/UI, however there are definitely limitations and external problems that aren't fixable at this stage. In future development, I'll be able to better address these bugs or add in requested features.